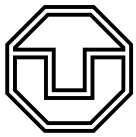


TECHNISCHE UNIVERSITÄT DRESDEN

**Faculty of Computer Science**, Chair of Privacy and Data Security

# Secure and Efficient Comparisons between Untrusted Parties

## Dissertation

Submitted to the Faculty of Computer Science, TU Dresden,
in Partial Fulfillment of the Requirements for the Degree of Dr.-Ing.

by
Dipl.-Inf. **Martin Beck**
born on 21 March 1984 in Eisenach, Germany

*Committee members:*

| | |
|---|---|
| Prof. Dr.-Ing. Thorsten Strufe | TU Dresden, Dresden, Germany |
| Prof. Dr.-Ing. Florian Kerschbaum | University of Waterloo, Canada |
| Prof. Dr.-Ing. Michael Schroeder | TU Dresden, Dresden, Germany |

*Day of defence:* May 17, 2018

Dresden, June, 2018

# Statement of Authorship

I declare that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Dresden, June 22, 2018

Martin Beck

# Abstract

A vast number of online services is based on users contributing their personal information. Examples are manifold, including social networks, electronic commerce, sharing websites, lodging platforms, and genealogy. In all cases user privacy depends on a collective trust upon all involved intermediaries, like service providers, operators, administrators or even help desk staff. A single adversarial party in the whole chain of trust voids user privacy. Even more, the number of intermediaries is ever growing. Thus, user privacy must be preserved at every time and stage, independent of the intrinsic goals any involved party. Furthermore, next to these new services, traditional offline analytic systems are replaced by online services run in large data centers. Centralized processing of electronic medical records, genomic data or other health-related information is anticipated due to advances in medical research, better analytic results based on large amounts of medical information and lowered costs. In these scenarios privacy is of utmost concern due to the large amount of personal information contained within the centralized data.

We focus on the challenge of privacy-preserving processing on genomic data, specifically comparing genomic sequences. The problem that arises is how to efficiently compare private sequences of two parties while preserving confidentiality of the compared data. It follows that the privacy of the data owner must be preserved, which means that as little information as possible must be leaked to any party participating in the comparison. Leakage can happen at several points during a comparison. The secured inputs for the comparing party might leak some information about the original input, or the output might leak information about the inputs. In the latter case, results of several comparisons can be combined to infer information about the confidential input of the party under observation. Genomic sequences serve as a use-case, but the proposed solutions are more general and can be applied to the generic field of privacy-preserving comparison of sequences. The solution should be efficient such that performing a comparison yields runtimes linear in the length of the input sequences and thus producing acceptable costs for a typical use-case. To tackle the problem of efficient, privacy-preserving sequence comparisons, we propose a framework consisting of three main parts.

**a)** The basic protocol presents an efficient sequence comparison algorithm, which transforms a sequence into a set representation, allowing to approximate distance measures over input sequences using distance measures over sets. The sets are

then represented by an efficient data structure — the Bloom filter —, which allows evaluation of certain set operations without storing the actual elements of the possibly large set. This representation yields low distortion for comparing similar sequences. Operations upon the set representation are carried out using efficient, partially homomorphic cryptographic systems for data confidentiality of the inputs. The output can be adjusted to either return the actual approximated distance or the result of an in-range check of the approximated distance.

**b)** Building upon this efficient basic protocol we introduce the first mechanism to reduce the success of inference attacks by detecting and rejecting similar queries in a privacy-preserving way. This is achieved by generating generalized commitments for inputs. This generalization is done by treating inputs as messages received from a noise channel, upon which error-correction from coding theory is applied. This way similar inputs are defined as inputs having a hamming distance of their generalized inputs below a certain predefined threshold. We present a protocol to perform a zero-knowledge proof to assess if the generalized input is indeed a generalization of the actual input. Furthermore, we generalize a very efficient inference attack on privacy-preserving sequence comparison protocols and use it to evaluate our inference-control mechanism.

**c)** The third part of the framework lightens the computational load of the client taking part in the comparison protocol by presenting a compression mechanism for partially homomorphic cryptographic schemes. It reduces the transmission and storage overhead induced by the semantically secure homomorphic encryption schemes, as well as encryption latency. The compression is achieved by constructing an asymmetric stream cipher such that the generated ciphertext can be converted into a ciphertext of an associated homomorphic encryption scheme without revealing any information about the plaintext. This is the first compression scheme available for partially homomorphic encryption schemes. Compression of ciphertexts of fully homomorphic encryption schemes are several orders of magnitude slower at the conversion from the transmission ciphertext to the homomorphically encrypted ciphertext. Indeed our compression scheme achieves optimal conversion performance. It further allows to generate keystreams offline and thus supports offloading to trusted devices. This way transmission-, storage- and power-efficiency is improved.

We give security proofs for all relevant parts of the proposed protocols and algorithms to evaluate their security. A performance evaluation of the core components demonstrates the practicability of our proposed solutions including a theoretical analysis and practical experiments to show the accuracy as well as efficiency of approximations and probabilistic algorithms. Several variations and configurations to detect similar inputs are studied during an in-depth discussion of the inference-control mechanism. A human mitochondrial genome database is used for the

practical evaluation to compare genomic sequences and detect similar inputs as described by the use-case.

In summary we show that it is indeed possible to construct an efficient and privacy-preserving (genomic) sequences comparison, while being able to control the amount of information that leaves the comparison. To the best of our knowledge we also contribute to the field by proposing the first efficient privacy-preserving inference detection and control mechanism, as well as the first ciphertext compression system for partially homomorphic cryptographic systems.

# Contents

# List of Tables

# List of Figures

# 1. Introduction

We are living in the age of information and more specifically just entered the "big data" era. The amount of data produced every year grows at a high pace. Two average days in August 2010 produced around five exabytes ($10^{18}$ bytes) of data worldwide, which is roughly the same amount generated from the dawn of men through 2003 [Kir10]. This amounts to about a zettabyte (ZB) (= $10^{21}$ bytes) of generated data for 2010. At the end of 2015, the annual global IP traffic should also reach the zettabyte barrier [Bar11]. The exponential growth in data transmission today is mainly driven by user generated content, Video on Demand (VoD) and increasing numbers of mobile devices, sensors and Machine-to-Machine (M2M) communication, which all can be — thanks to technological progress and Moore's law — produced at lower costs and higher quality than ever before. The same technological advancements of course also drive research areas, like astronomy, physics or genomics. Beginning with a report in 2001 [Lan01], issues, challenges and solutions to the rapid data growth are often discussed under the "big data" term.

The digital universe is growing rapidly and this trend is expected to continue. In 2012 the digital universe comprised 2.8 ZB (4 ZB in 2013) and is projected to cover 40 ZB at the end of 2020 as estimated by Gantz and Reinsel [GR12]. They came to the conclusion that 23% of the information within the digital universe could be useful, but only 0.5% are actually analyzed.

Hand in hand with the technological challenges on how to transfer, process and store such large and ever growing amounts of data, goes the task of ensuring privacy of the affected users. While users can more or less directly influence what Personally Identifiable Information (PII) within user generated content is shared online, the same control is less obvious — if not impossible — for areas such as eHealth, recommender systems or governmental data processing. The fraction of the digital universe that needs to be protected via security or privacy measures is expected to grow, however, nearly half of the respective data is unprotected [GR12]. Current issues with massively centralized data storage are regularly covered in news reports. Data breaches and leaks possibly affect hundreds of millions of people around the world each year [Ide15]. Over the last 10 years, the "Identity Theft Resource Center" recorded 6079 breaches exposing in total more than 862 million

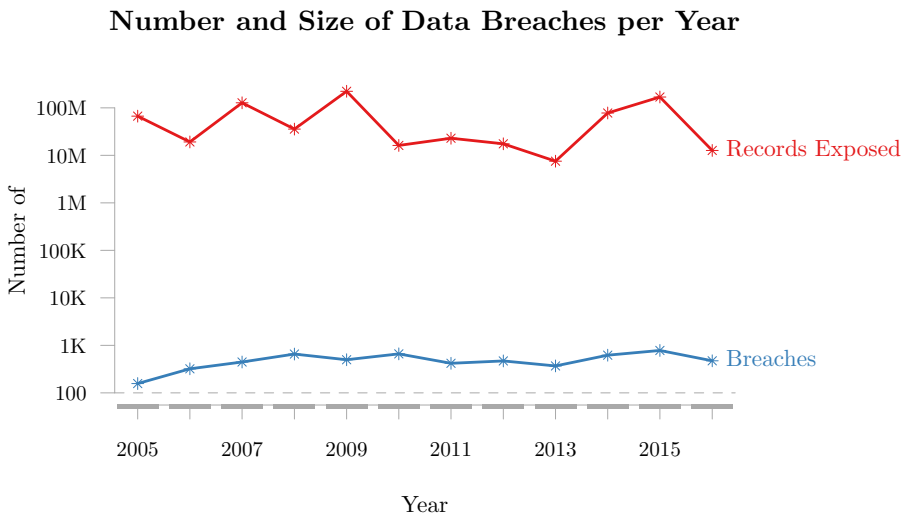



Figure 1.1.: Number of data breaches from 2005 till June 2016 together with the accumulated amount of exposed records for all breaches. Source: Identity Theft Resource Center [Ide15]

records. Figure 1.1 shows the number of breaches and the number of exposed records per year from 2005 to June 2016.

The gigantic magnitude of affected users represents the need to find solutions to security and privacy problems on multiple domains. Legislation must find ways to adopt to the fast changing conditions due to technological advancements and more importantly deal with data and organizations spread across different countries. Society at large has to gain competence in securely using the new technology, judge possible implications of digital actions and understand that the sense of anonymity might be elusive. Researchers and engineers must find ways to protect, preserve and enhance privacy, anonymity and in general security for all users and build systems that make transparent use of cryptography or similar techniques to protect sensitive information by default.

Therefore specific systems that are efficient, effective and privacy-preserving are highly anticipated to enable confidential data exploration and utilization while giving provable privacy guarantees. One elementary part of such constructions is the ability to store and provide information as necessary. This can be accomplished using databases, web services or specific transfer protocols. However, any of these techniques may — and for larger collections of data must — provide ways of filtering, searching and comparing entries for the required or most relevant information quickly.

As long as a query towards such a system, as well as the stored data does not contain any PII or is by any other means confidential, it might be processed without any privacy preservation, using very efficient techniques for filtering, comparison and search. However, once the confidentiality of either must be preserved, similar efficiency for the same functionality is much harder to achieve. Many different solutions can be found in the relevant literature to perform different kind of searches and comparisons for various security/privacy goals and attacker models. Therefore we will describe a possible use case which fits the overall problem scenario, sketch open questions and elaborate on sensible properties for a desired solution.

## 1.1. Setting

The central theme, which will always reoccur throughout this thesis is DNA-sequencing and it therefore serves as a use case to motivate and align objectives and desirable properties to. The reason for looking at genomics comes from the immense technological improvements that happened over the last decade. In particular the introduction of high-throughput sequencing combined with the quest for sequencing a whole human genome for less than $ 1000 drove sequencing costs down so far, that a lot of sequenced genomic data was collected and is available through several online databases. Figure 1.2 shows the cost development for sequencing a single human genome over the last 14 years in relation to what Moore's law predicted. The "1000 Genomes Project" sequenced 1092 genomes and made them available [Abe+12]. Genomics England, a company owned by the Department of Health in Great Britain is coordinating the "100.000 Genomes Project", which should have 10.000 genomes sequenced in a pilot phase at the end of 2015 and should be finished in 2017. The costs for this huge sequencing project are estimated at around £ 300 million [Rab14].

An ultimate goal derived from the extracted knowledge of thousands of genomes is the establishment of personalized medicine. Personalized medicine describes individual treatment for every patient depending on the genomic predispositions and current condition to select the correct drug and dosage for an optimized treatment plan. Of course personalized medicine does not only depend on the availability of a large number of sequenced genomes, but also on radical changes on how drugs get approved to be safe and provide real health benefits. The current approval process conducts empirical measurements of different dosages over larger groups, which cannot be applied anymore, due to the individual drug selection and dosage.

At some point, patients might carry their sequenced genome, health record and related information around on a smart card-like device. Doctors and emergency physicians can then query this information to check whether a patient received

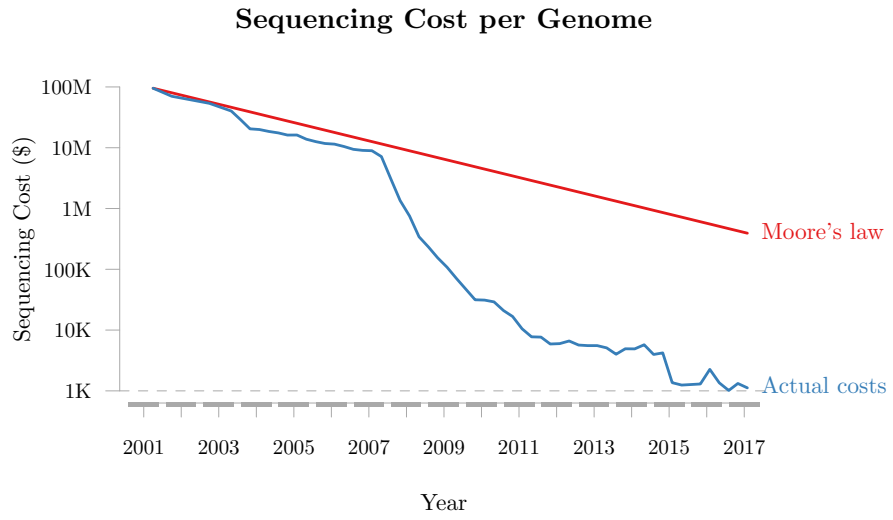



Figure 1.2.: Evolution of genome sequencing cost from 2001 till 2015 in contrast to the prediction made by Moore's law. Source: Wetterstrand [Wet15]

some specific inoculation, having allergies, needs specific medication or has genetic predispositions, which point to some intolerance prohibiting the use of certain drugs. As all of the mentioned data contains a lot of personal information, it is in the interest of the patient to protect this information carefully.

Privacy protection is also required by law in many countries. As such Germany states in the Bundesdatenschutzgesetz (BDSG)[1] that collection, processing and use of private information should be restricted to a minimum. In particular, data can only be used for the specific purpose it was collected for and should be anonymized as far as possible (§3a BDSG). A person can of course always give his informed consent to have private data used for further purposes. As long as the data subject (person) was informed about the purpose of the collection, processing and use, its consent instantly resolves privacy issues from the standpoint of jurisdiction (§4a BDSG). However, if a data subject doesn't want to give consent to a disclosure, but still wants to use a service that requires private information, specific privacy-preserving mechanisms are necessary to utilize the personal data for provision of service.

Following the personalized medicine example, a patient might want to let the doctor learn necessary information for medical decisions, but nothing more about

[1]The Bundesdatenschutzgesetz (BDSG) is the German Federal Data Protection Act (FDPA). We refer to the Federal Data Protection Act in the version promulgated on 14 January 2003 (Federal Law Gazette I p. 66), as most recently amended by Article 1 of the Act of 25 February 2015 (Federal Law Gazette I p. 162)

his genomic data. The patient could disclose genetic information, but may rather wish to be treated properly without the necessity to entrust another party with private information. Furthermore, disclosed data may of course still contain PII and therefore fall under prevailing law.

We describe the setting in which we perform our research and argue why the corresponding choices were made. The desired overall functionality of the system is to allow an efficient comparison of two private elements, each hold by a different party. Each party of course does not enjoy complete trust from the respective other. Privacy-preserving techniques are used to preserve confidentiality of all private inputs. Only the final result is given to the initiating party. The result might be an approximation of a distance measure. A more formal description is given in the following paragraph. For ease of presentation the initiating party is called "user", while the other party is termed "server".

Given a user $a$, holding a text string $s_a \in A^*$ over alphabet $A$ and a (database) server $b$, holding a text string $s_b \in A^*$, the task is to calculate the distance of both strings regarding some metric $d\colon A^* \times A^* \to \mathbb{R}$ over text strings and return an approximation of $d(s_a, s_b)$ while respecting the privacy of $a$ and $b$. The Levenshtein distance [Lev66] — also known as edit distance — might serve as the mentioned metric $d$. However, it can be any other metric and indeed also over a different domain. The limitation on possible metric spaces $(D, d)$ is the existence of an embedding with low distortion such that $f\colon D \to \{0,1\}^m$ maps elements from $D$ to binary strings of identical length $m$ and the target distance metric is the Hamming distance $d_H$, as described in Section 1.5. This requirement can be relaxed slightly as low distortion embeddings from several domains into the Hamming distance binary string metric space $(\{0,1\}^m, d_H)$ are known.

Du and Atallah [DA01] separates four models for query-privacy as depicted in Figure 1.3. The can be characterized as follows:

(a) Private Information Matching (PIM) describes the model in which Alice has a confidential string and wants to privately query Bob's private database. Neither Alice nor Bob should learn more than necessary.

(b) Private Information Matching from Public Database (PIMPD) can be seen as a relaxation of PIM, as Bob's database is public and therefore the reply may leak additional information.

(c) Secure Storage Outsourcing (SSO) presents the secure outsourcing scenario. It allows Alice to use a less-trusted external party with potentially more resources to perform confidential tasks for her.

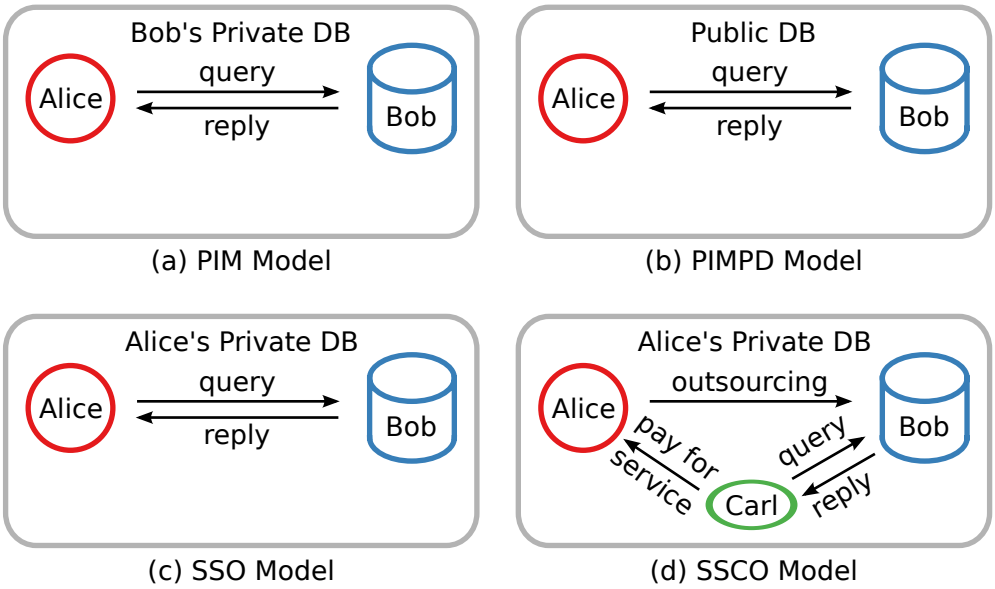



Figure 1.3.: Models for private queries. Source: Du and Atallah [DA01]

(d) Secure Storage and Computing Outsourcing (SSCO) is similar to PIM, it just includes a third party in between Alice and Bob, which is handling the requests. Such multi-party schemes often require non-collusion between any pairs.

The approach described in this dissertation fits best into the PIM model (a).

**Thesis Outline** The next section (1.2) will therefore describe the detailed requirements, why they are reasonable and what these requirements imply. It follows a discussion of open research questions (Section 1.3), which are then further explained together with a brief outline of the main contributions of this thesis in Section 1.4. Section 1.5 introduces important concepts that are necessary for a better understanding of the following chapters. Finally, Section 1.6 contrasts the topics and concepts that are considered within this thesis against related problems that are out of the scope of this work.

## 1.2. Requirements

Following the issues presented over the previous pages, a list of requirements can be extracted that an appropriate system must fulfill. These requirements are listed

and briefly introduced below. It follows a more detailed discussion for the chosen requirements including their justification.

**Privacy-preserving for all parties**
: Confidentiality for input data

**Efficiency for long strings**
: Comparing long sequences efficiently, preparing comparisons of genomic sequences

**Approximate matching for similar strings**
: Return distance approximation or match similar strings within a certain distance

**Non-interactive protocol**
: Minimize interactivity of protocol to support asynchronous communication and to allow unreliable mobile network connections

**Two-party**
: Minimize involved parties to reduce overall attack vectors and also risk of collusion

**Resource-constraint mobile clients**
: Distribute the mobile client work load and shift it towards the server, without overloading the server

**Size-hiding**
: Leaked information about the size of the client input should be rather low

**Inference control**
: Detect and reject queries that belong to an inference attack

These requirements will be used to assess the relevant literature, guide own constructions and drive the relevant evaluation parts. The following sections will describe requirements in more detail and motivate their use.

### 1.2.1. Privacy-Preservation

A reason for using a privacy-preserving protocol in general was already given at the beginning of the chapter. If the goal is to evaluate a certain function $F$ over private input $x, y, z$ from different parties (or from a single party via outsourcing) it is sufficient for appropriate parties to learn $F(x, y, z)$ and nothing else. It is especially not necessary for a party to learn anything else than the result of $F$ and its own input. No intermediate results should therefore be leaked. Appropriate parties are those inputting own private data. It is in their best interest to protect this data.

So in a two-party, distance evaluation protocol, as it is described in Chapter 4, a client should learn nothing but the result of $F$ — the actual (approximated) distance. The server should learn nothing about the client query, even after successive queries. So both parties can express and enforce their security and privacy requirements, which is called *multilateral security* [Ran00].

Privacy preservation is not the only goal, as otherwise parties would not even engage in a joint/secure computation. It follows that privacy must somehow be balanced with other goals, which are mainly functionality and performance in our case. Furthermore, privacy is a protection goal next to others like authenticity and availability. For these goals meaningful attackers must be defined against which the proposed solution should be secure. The attacker description will be discussed in Section 1.5.1.

### 1.2.2. Efficiency for Long Inputs

We especially want to support efficient comparisons of long strings or sequences. "Long" in this case refers to strings with more than $10^4$ subsequent characters from the alphabet for a single string. Fulfilling this requirement allows privacy-preserving comparison of long sequences, as it can be found in the genome matching use case, where strings of such length — somewhere between $10^4$ and $10^9$ characters, depending on the actual task — must be compared against each other.

The overall comparison efficiency should be high, which means that the compute time should scale linearly with the length of the longest input. We embed the string distance function into the Hamming space and use the Hamming distance as an approximation. The length of the bit vectors representing the strings is linear in the length of the longest string. Calculating the Hamming distance through secure computation has complexity $\mathcal{O}(n)$ with $n$ being the length of the bit vectors. This follows from the fact that the bit vectors must be XOR'ed element-wise.

As such the efficiency is optimal for a protocol that calculates the exact Hamming distance and leaks no information to the executing party. A scheme cannot calculate the exact Hamming distance securely in sub-linear time. Furthermore, constants should be rather low to get to an overall practical solution. If a more space efficient embedding is used, which would transform strings into sub-linear length bit strings, then our protocol directly profits from these changes.

### 1.2.3. Approximate Matching

The problem setting of *approximate string matching* is chosen, as it naturally applies to many cases in which strings are to be compared or searched that might contain some sort of error. Examples of such errors include typographical errors or spelling mistakes, noise from communication channels and genomic sequencing errors. Thus an approximate matching algorithm can be used to identify such errors or find most similar elements. For the latter case the problem of approximate matching is closely related to the nearest-neighbor search, which specifies the problem of efficiently selecting the most similar elements to a given query regarding some distance metric.

Two different uses for approximate matching can be found in the literature.

1. Deciding whether two or more inputs are pair-wise close to each other, *i.e.* their distances are below some threshold. The only answer will be a binary decision about being close or not.

2. Calculating or estimating a distance between inputs and returning this (approximated) distance.

In the first case the matching algorithm should return a binary answer and as such an approximate match is found if the distance between the query and the compared element is within a given threshold range. The second case implies that the algorithm returns a distance or similarity value of some sort describing how close the compared elements are.

This thesis uses and supports both notions, which will be detailed and separated again by the presented protocols in Chapter 4. In general, if some similarity measure is generated, it can either be given to the client directly or used for *in-range* testing. The result of this test is the desired binary answer, which is used as output to the client. However, as required by privacy preservation (see Section 1.2.1), neither the actual resulting distance, nor the result of the in-range testing must be leaked to the server. Also following the privacy requirement: In case the actual distance is unnecessary for the client to learn and a lower entropy answer would be enough, the in-range check should be chosen.

The choice of distance or similarity measure is not fixed and relies upon the used methods, the overall comparison protocol and available embeddings between relevant metric spaces. Equally, selecting the correct notion of approximate matching must be decided upon the actual use case.

### 1.2.4. Non-Interactive Protocol

When two or more participants communicate via some protocol, they exchange messages in a mostly predefined way. An interactive protocol in our case then means that a party $A$ receives input from another party, which depends on a previous input from $A$. Furthermore, the previous input must belong to the same protocol execution and the party $A$ must output something depending on its recent input. Following this understanding, information in an non-interactive protocol would pass each party only once, except if the party is a sink for the mentioned information.

Relevance follows from the resource-constraint and possibly mobile clients (see Section 1.2.6) and their possibly rather unreliable communication channel. The influence of interactivity on the overall runtime and latency potentially increases with the number of interaction rounds. Network interruptions are also less critical in the non-interactive protocol setting due to the infrequent and possibly shorter communication times.

Furthermore, a non-interactive protocol can easily be performed via asymmetric communication, such that no two parties must be online at the same time. In the two party case, as described in this thesis, this would imply posting a comparison request somewhere so the other party can later on fetch it and initiate the comparison. The result would again be posted for the requester to be collected. The same asymmetric communication can of course be used for interactive protocols, but each additional interaction round potentially introduces delays due to the latency in posting and fetching messages. A non-interactive protocol is therefore better suited for asymmetric communication than a highly interactive one.

### 1.2.5. Two-Party

We will focus on a *two-party* setting and more specifically a client-server scenario in which a client has an input string upon which a distance metric should be approximated. We could integrate more parties with pairwise exclusive knowledge and thus privacy requirements into the protocol, but this would either increases the necessary trust in parties to not take part in collusion attacks, or require

strengthened protocols withstanding such attacks. Both drawbacks are not desired, so unless it is noted otherwise, we will be working in the two-party setting.

The two-party setup is a natural fit, as we could have a patient on the one side as a user, possibly using a low-power and bandwidth-constrained device like a smart card, and the physician with a larger genomic database represented by the server. The server actually stores genomes or snippets of them, which are characteristic for a certain predisposition and thus act as representatives for that class. The task would be to find the most similar entries in the database to learn possible predispositions, or to match classes of problematic genomic mutations.

### 1.2.6. Resource-Constraint Clients

Under the assumption that a client initiating a comparison might be a resource-constrained device, *e.g.* a mobile client, it is necessary for such devices to have algorithms that use as few resources as possible in order to reduce power consumption and minimize memory usage. The overall goal then of course is to maximize their battery life. Furthermore, due to limited available resources, such devices might not even be able to handle asymmetric cryptographic operations (within practical runtimes) and thus be able to participate in such a secure computation at all. Examples for computation times of asymmetric and specifically homomorphic operations on smart cards can be found in the relevant literature, *e.g.* from Bichsel et al. [Bic+09]. This requirement also includes the workload distribution between the client and server. However, even under these restrictions we want such a *thin client* to participate in the overall protocol and input its private data securely.

### 1.2.7. Size-Hiding

The requirement of hiding the size of the input extends the privacy requirement and is mentioned as an explicit requirement and construction goal, as many related work proposals are not addressing this issue, even though they preserve privacy in general. In particular, it helps in cases when an outsider or a participating party might be able to infer sensitive information from the length of another input. An example: If the input to the system is either “yes” or “no” and the pre-processing steps (including encryption) do not hide the size or in this case length of the input, then an adversary can easily distinguish ciphertexts and break the confidentiality — even in case the underlying cryptographic scheme is semantically secure. As we will be working with operations over sets and set sizes carry valuable information, we thus also require that the proposed scheme has a size-hiding property.

### 1.2.8. Inference Control

Any protocol must return information or have an output of some sort. This might be either in form of a direct result or an intermediate result for further operations. The same is of course also true for privacy-preserving protocols. Considering the privacy-preserving evaluation of a distance metric between two parties, each having a sensitive input, one of the participants can infer knowledge about the other party's input by repeatedly asking for distance calculations of slightly different inputs. An adversary could for example perform an adaptive attack in which he checks the distance result from the previous queries and adopts his next input to reduce the actual distance between his own input and the other party's. Once he reaches a distance of zero, he learned the confidential input from the other party. Such a greedy algorithm can uncover a private input very fast (with very few steps). The actual number of steps of course depends on the metric used, the inputs and the amount of information that is returned by the privacy-preserving protocol.

Secure computation schemes therefore only guarantee that private inputs from other parties are kept confidential during execution and without learning the result. Protection from inference attacks upon the output cannot be achieved this way. To circumvent this, anonymization schemes like differential privacy [Dwo08] were developed to restrict the information that is contained in the (published) results. Several issues exists when dealing with countermeasures against inference attacks, as will be described in Chapter 5, however, we still want to be able to influence the success rate of inference attacks, while results should still carry enough information to be usable.

## 1.3. Open Research Questions

The relevant literature on describing comparison schemes is rich. They are based on different methods to achieve privacy-preservation, allow different functionality, are secure against different adversaries or deliberately choose low security levels to be as efficient as possible. Chapter 2 will group them into classes of similar methodology and reviews the properties of these classes with respect to the above stated requirements. However, even though there is a wide range of work on comparison schemes and we picked the ones most fitting our requirements, there are still unanswered questions, which will form the basis of all further research conducted and presented throughout the rest of the thesis. It starts with rather generic questions, which lead to the foundation used within further, more specific research.

(1) How can an efficient, privacy-preserving, approximate comparison scheme be constructed?

(2) How can an ongoing inference attack be detected and limited, given encrypted requests?

(3) How can a client reduce cryptographic overheads and transmission bandwidth?

A description for the single parts of the privacy-preserving comparison system is given to outline the thesis and introduce the contributions.

### 1.3.1. Efficient Approximate String Matching

As introduced above, the overall comparison system is a composition of several different schemes, each having a specific focus. We start by describing an efficient and effective protocol for secure approximation of a metric over the input of two parties. Chapter 4 will propose this core comparison scheme, instantiated to compare character sequences. When two parties, each holding a string, want to approximate a string similarity of their sequences efficiently, they must choose an appropriate measure. The Levenshtein distance is therefore used as an example for the necessary norm to be approximated. We present an embedding to first map long strings to sets using character grams, then map the built sets to set representations in the form of binary strings. Upon these, set operations estimate the symmetric difference and together with a final cardinality estimation approximate the initial metric over strings. We show that the approximation is highly correlated to the original metric and that the scheme is privacy-preserving for an Honest-But-Curious (HBC) querier and a malicious server. The scheme is implemented, evaluated and tested using the Human Mitochondrial Genome Database (mtDB).

### 1.3.2. Inference Control for Confidential Queries and Answers

Every privacy-preserving protocol must output some information in the end, otherwise it would have no use. An attacker can leverage this information to gain knowledge about the confidential input of another participating party. One can describe a game in which the attacker crafts input to the protocol in a way to generate high entropy results for him. Winning the game means reconstructing the secret hold by the other party. This game and applied technique is very similar to the board game “Mastermind” (depicted in Figure 1.4), which served also for an efficient attack on genome sequence comparison protocols [Goo09].


Figure 1.4.: A game of Mastermind. Source: ZeroOne [Zer05]

By design, such inference attacks make use of rather close queries to learn detailed information about specific parts of the input. We hereby describe a novel technique to detect such close queries against a privacy-preserving protocol. The proposed construction can directly be used against Mastermind-like attacks, which draw conclusions by submitting many very close queries and evaluate the returned results. Due to the close-query property of such attacks, closeness detection is used for the inference control algorithm to detect and drop too similar requests. We further generalize the Mastermind attack to also work with incomplete and rejected protocol invocations — as they might happen with inference control mechanisms in place. We show that due to the constructions used in the core similarity matching protocol from Chapter 4, the algorithm can effectively separate Mastermind-like attacks from genuine close queries. Furthermore, incorporation of specialized Zero Knowledge Proofs (ZKPs) allows us to prove the protocol to be secure in the malicious-prover, honest-verifier setting.

### 1.3.3. Secure Computation for Thin and Mobile Clients

As the ideas towards personalized medicine suggested energy-constrained small, smart devices to carry confidential information, this information must be utilized somehow. It cannot simply be handed over to other devices for remote computation, due to being highly private information. Further, computation will also not be carried out directly on these constrained devices, as they might be too weak to perform complex tasks. Therefore, computation might be outsourced to more capable, possibly centralized compute nodes while keeping the overall work load of

the client side rather light. This asymmetric requirement on computational power will be worked on in Chapter 6. Another constraint that comes with small and mobile devices is their network connection in terms of throughput, stability and availability. As all of these properties will be of rather low quality — compared to a wired network connection — it is important to keep the bandwidth overhead introduced by the privacy-preserving protocol as low as possible.

The latter point is mainly problematic due to fact that asymmetric encryption systems and most homomorphic encryption systems in general come with a high expansion factor when comparing plaintext to ciphertext sizes. This problem can be circumvented by using symmetric cryptography to encrypt data coming from the smart device. The natural question that arises is: How to translate the symmetrically encrypted plaintext into a homomorphically encrypted plaintext on the server side? Recent proposals suggested to encrypt the symmetrically encrypted data homomorphically and run the symmetric decryption algorithm using the homomorphically encrypted symmetric key within the homomorphic encryption system. This allows for small transmissions, but due to the necessary use of a Fully Homomorphic Encryption (FHE) or Leveled Homomorphic Encryption (LHE) system for the scheme transition, it enforces a huge computational overhead on the server side, completely defeating the communication benefits achieved in the first place.

To solve this problem, a novel use of PRNGs and Partly Homomorphic Encryption (PHE) systems is presented, achieving very low transmission overhead together with dramatically minimizing the computational overhead on the server side compared to all previously proposed scheme conversion solutions. The system is further analyzed regarding its security, performance and generalization of the core idea. A variant is described in which the symmetric keystream computation is outsourced to a trusted and more powerful device, like a smartphone, laptop or desktop PC. This variant also achieves a significant decrease in computational overhead on the client side, which is desired for the asymmetric distribution of computational capabilities.

## 1.4. Contributions

This dissertation contributs to the research area in various ways. The following contributions describe general achievements, while detailed contributions in specific domains are described in the relevant chapter.

1. An *efficient privacy-preserving approximate string matching scheme*, which estimates the Levenshtein distance between two confidential strings belonging to different parties with low distortion for small distances. The initiator

learns the approximate distance or that the approximate distance is within a predefined range, while the other party learns nothing. The scheme is shown to be secure in the HBC setting. The detailed description follows in Chapter 4, while the core results are published in [BK13].

2. A protocol for *privacy-preserving similar input detection*. A binary output describes if a private input is closer than a certain threshold to all previous inputs or not. This 1 Bit information is used to detect and control inference attacks, which can otherwise be used to efficiently infer knowledge about the confidential information hold by one of the participants. A detailed analysis demonstrates its applicability. This is to the best of our knowledge the first scheme that tries to detect and mitigate inference attacks while working only with encrypted data. Chapter 5 gives the details of the protocol extensions, with core findings being published in [KBS14].

3. A scheme for *homomorphic ciphertext compression*, which combines the efficiency of symmetric encryption and the functionality of homomorphic encryption. A novel way to construct a symmetric stream cipher based on the homomorphic decryption function is presented, which leads to a highly efficient transition scheme. Thus, encrypted data sent from the client to the server, can be transformed from data encrypted under a symmetric stream cipher into the same data being encrypted under a homomorphic cryptographic system. The server can then perform ordinary secure computation upon it. Overall this enables a resource and bandwidth-constrained device, like a mobile phone, wireless sensor node or smart card to securely outsource computation with low encryption and transmission overhead. Again, to the best of our knowledge we are the first to describe such an efficient scheme, with a transition function that is several orders of magnitude faster than comparable proposals. The detailed scheme is described in Chapter 6, while results are published in [Bec15].

## 1.5. Preliminaries

Constructing a comparison scheme requires the definition of an appropriate similarity measure by which elements are compared. Finding a good metric (or a function using different metrics) that resembles the desired similarity measure may by itself be a non-trivial task. However, it is assumed that this step is already done and the measures to be approximated by the comparison scheme are known and specified. We will introduce definitions for metric, metric space, closeness and embeddings, as these are basic primitives, which are necessary for the construction of a system that compares elements.

As denoted in Section 1.5.2, we use the terms distance, similarity and closeness interchangeably throughout the document. A metric however is clearly defined as a function $d\colon D \times D \to \mathbb{R}$ over a set $D$ that satisfies the following axioms for $x, y, z \in D$:

1. $d(x, y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$,
2. $d(x, y) = d(y, x)$
3. $d(x, y) + d(y, z) \geq d(x, z)$

The combination $(D, d)$ is called a metric space. The function $d$ defines the distance between elements in $D$. On the other hand, a similarity function $s\colon D \times D \to \mathbb{R}$ should specify the amount of similarity between elements in $D$. As a common definition or necessary axioms for a similarity function do not exist and typical similarity functions are some kind of inversion of the associated distance, we state that a similarity function can be constructed from any given metric by $s(x, y) = \frac{1}{1+d(x,y)}$.

Upon a metric $d$ the distance between an element $x \in D$ and a set $A \subseteq D$ is defined as the infimum $d(x, A) = \inf_{a \in A} d(x, a)$. The element $x$ is called close to set $A$ if $d(x, A) = 0$.

The final comparison scheme receives a query $q$, compares it to an element $y$ and outputs a binary result of whether an approximation of a distance $d(q, y)$ is close to a certain value or not. If we take the previous definition of closeness, the comparison scheme would only return *true* for elements $q, y$ that are identical. However, the scheme should perform fuzzy matching to reveal also highly similar elements that are not identical $d(q, y) \leq t$, having an approximated distance score equal to or below some threshold $t$. We therefore require two elements $a, b \in D$ to have a distance $d(a, b) \leq t$ to be similar.

Metric spaces can be compared and sometimes translated from one to another. For comparing two metric spaces $(A, d)$ and $(A', d')$, a function $f\colon A \to A'$ is used to map elements from $A$ to $A'$. This map is called an embedding. If this map preserves the distance, that is $d(x, y) = d'(f(x), f(y)) \quad \forall x, y \in A$, it is called isometric. If such a function exists for a metric space $(A, d)$ towards the metric space $(\mathbb{R}^k, d_p)$, with some $k$ and $d_p$ being the $l_p$-distance and thus $d(x, y) = d_p(f(x), f(y)) = \|f(x) - f(y)\|_p \quad x, y \in A$, then $d$ is called an $l_p$-metric.

If isometric embeddings cannot be found for two metric spaces, most often a map is used that results in similar distance functions $d$ and $d'$. The similarity between the distance functions can be evaluated by calculating maximum factors

for stretching and shrinking the distance values $d(x, y)$. The maximum shrink factor, or *contraction* is given by

$$\max_{x,y \in A} \frac{d(x, y)}{d'(f(x), f(y))},$$

whereas the maximum stretch factor, or *expansion* is given by

$$\max_{x,y \in A} \frac{d'(f(x), f(y))}{d(x, y)}.$$

The *distortion* of $f$ is the product of the contraction and the expansion. Embeddings will be used later on in Section 4.2 to translate between character strings distances, set similarity measures and binary string distances.

### 1.5.1. Adversary model

Within the previous sections comments regarding the strength of the adversary were made, without ever actually defining possible attackers. It was left open to describe meaningful adversaries, against whom the comparison scheme should be secure. Defining such attackers involves many parameters, which influence the overall abilities and strength of an attacker. The following list describes some of the dimensions that should be thought about together with some examples or a brief description.

**Role**
Examples are: Vendor, Developer, Administrator, User, Outsider

**Area of physical control**
What is physically accessible by the adversary?

**Follows the role restrictions**
Distinction in Honest-But-Curious (HBC), Covert, Malicious.

**Time, Money**
Most often treated as a factor on estimating practical security.

**Computational power**
Unlimit computational power implies an information-theoretic adversary.

**Knowledge**
The adversary could have knowledge that is not publicly known.

**Active vs. Passive**
Does the attacker participate and actually send information?

**Adaptive vs. Non-Adaptive**
Distinguishes attackers that adapt their strategy based on previous results.

Time, money and computational power influence each other, as the computational power changes according to the amount of money and time that is available. If there are no restrictions on the computational power, we call the attacker an information-theoretic attacker and computational-complexity-restricted otherwise. To increase the strength of a cryptographic system against this property, higher entropy or longer keys are used. Similarly the area of physical control is influenced by the role an adversary has.

The following sections will discuss the typically distinction used on the self-restriction of adversaries for cryptographic protocols. These are modeled most often using the HBC and malicious model. Another rather new model in between those two is the covert adversary.

### Honest-But-Curious (HBC) Model

From the efficiency requirement in Section 1.2.2 directly follows an obvious candidate for the attacker model. We are therefore mainly considering an *computational complexity* restricted attacker within the HBC model. Such attackers are assumed to have certain limits in what they can do. They want to learn as much as possible by *observing* all information they can get during the participation and execution of a protocol. So first of all they are allowed to:

- Evaluate arbitrary deterministic polynomial time (PTIME) algorithms.
- Look at the inner state of all algorithms run by himself.
- Use all input to and from himself.

Next to these granted rights, there are also limits. He is assumed to comply with:

- Follow the given protocols.
- Run algorithms correctly.
- Use correct inputs.
- Return correct answers.

HBC adversaries are also allowed to take a complete transcript of the whole protocol execution for later analysis, however, they are especially assumed to not disclose or share secret information. Collusion is therefore explicitly excluded — which is not relevant as we will design a two party scheme, following our requirements (see Section 1.2). Furthermore, we follow the definitions of a semi-honest and malicious model from Goldreich [Gol04].

If we would allow an attacker to solve problems from a complexity class above PTIME, theoretical proofs for secure protocols and algorithms assuming PTIME attackers would not hold anymore. This is a realistic assumption, as an attacker not bound to PTIME algorithms, for example being able to evaluate arbitrary deterministic exponential time (EXPTIME) functions, can solve all problems in EXPTIME and trivially also in non-deterministic polynomial time (NP), easily — which is assumed to be hard for many security proofs.

Further, if we would consider an computationally unbounded attacker, only information-theoretically secure cryptosystems and therefore just private key cryptography, like the one-time pad for confidentiality and the information-theoretically secure Message Authentication Code (MAC) for authentication and integrity could be used. Public key cryptography would not provide any security against such an attacker.

### Malicious Model

If a protocol participant is unlikely to follow the HBC model due to strong own interests in gaining information, privileges or other advantages, the overall protocol construction must account for and withstand the expected behavior. If a party is expected to be HBC in general, but might deviate at a few, specific or critical points in the protocol, these places can be protected using a few efficient ZKPs to verify correct behavior of the addressed party. We use this technique in the inference attack detection framework in Chapter 5.

Lifting the attacker in general from the HBC into the malicious model would require protocols secure against malicious attackers. Such attackers might deviate arbitrarily from the protocol, sending false, forged or no answers at all. The necessary level of security can be reached by generic extensions that apply zero-knowledge proofs to constructions secure in the HBC model. As a result security in the malicious model [KK08] is achieved. In case secure computations based on Yao's garbled circuits [Yao86] is implemented and must be extended to be secure in the malicious model, cut-and-choose based methods can be employed [Lin13]. However, both extensions, ZKPs and cut-and-choose techniques imply large overheads in communication and computation, driving secure solutions far away from the previous requirement of having a highly efficient and practical system. We therefore keep the HBC and PTIME computational-complexity attacker model and possibly apply efficient ZKPs at necessary places.

#### Covert Model

Aumann and Lindell [AL09] introduce the notion of an *covert* adversary. The strength of this adversary lies between the well known HBC and malicious attacker models. The intention of the adversary itself is to cheat but only when the probability of being caught is below some threshold $\epsilon$. In this sense they define a simulation-based security notion for *covert adversaries with $\epsilon$-deterrent*. The guarantee is not about absolute privacy — a party can cheat and learn some private information, but will be caught cheating with probability $\epsilon$.

This model is often applied to secure computation protocols and more specifically Secure Multi-Party Computation (SMPC). It follows the idea of cut-and-choose protocols in which a certain function is evaluated more than once and randomly checked for proper evaluation. However, Aumann and Lindell [AL09] described attacks against cut-and-choose, which were fixed and lead to an improved construction.

### 1.5.2. Wording

To ease understanding of the following chapters a convention on the use of certain expressions follows. Terms which we use interchangeably throughout this document are presented and sketched. They will be differentiated at necessary places and differentiation will be made explicit.

**privacy-preserving, privacy-friendly, privacy-enhancing, secure**
: With respect to a protocol these adjectives refer to an effort made by the protocol or underlying algorithms to protect sensitive information of at least one participant.

**sensitive, confidential, private**
: Input or output data may be classified by such terms and implies that the respective party has an interest in protecting the contained information.

**cryptographic system, cryptographic scheme, cipher, encryption system**
: A system, scheme, protocol or algorithm that mainly uses cryptography to achieve certain properties like confidentiality or authenticity.

**cryptographically secure, computationally secure, indistinguishable**
: Cryptographic primitives are often designed to withstand adversaries with limited capabilities. The restriction in their power is typically described by allowing them to evaluate arbitrary deterministic polynomial time (PTIME) algorithms. Cryptographic primitives are called computationally secure if the probability of such an adversary in differentiating their output from a random variable following the same distribution is bounded by a value $\epsilon$. The output is then called computationally indistinguishable.

---

**distance, similarity, closeness**
A measure describing how close or similar two elements are, while distance not necessarily describes a distance metric (see Section 1.5).

**close, similar**
Two or more elements with a distance below or similarity above some threshold.

**(bit) vector, array, bit string**
Two or more bits with a specific order can be placed in a vector, array or concatenated into a string interchangeably.

**(character) string, character sequence**
Similar to the bit string described above a character string is a concatenation of character symbols from some alphabet.

---

**party, participant**
One of several attendees belonging to a protocol (see Chapter 4).

**client, user**
Party that initiates a comparison, assumed to be computationally weaker than the server (see Chapter 4 and 6).

**server, service provider, cloud**
Party that offers elements to compare with (see Chapter 4).

---

**attacker, adversary**
An external or internal algorithm, person or group interested at gaining more information than allowed (see Section 1.5.1).

**honest-but-curious, semi-honest, observing**
The attacker model used for most parts of the document (see Section 1.5.1).

---

**query, request**
Data sent from the client to the server.

**result, answer**
Data sent back from the server to the client.

## 1.6. Out of Scope

As the focus of this work lies upon construction and evaluation of methods for efficient and privacy-preserving element comparisons with not fully trusted parties, there are many closely or loosely connected topics to all the requirements and desired attributes. A few of these touching areas are mentioned below and refer to interesting topics, which are explicitly not studied. Solutions and proposals in these fields are taken as given, are mentioned to be used at appropriate places and inserted transparently as building blocks whenever needed. As such these fields will also not be discussed in general within the related work in Chapter 2 or over the following protocol construction chapters. If necessary, proposed solutions might deviate slightly on the exclusion of the following topics, but whenever this might be the case, it is clearly stated and made explicit.

**Network** One of the main overall goals is to enable two parties to jointly process data. Therefore a necessary requirement for those two parties is of course to be able to communicate with each other. We do not care about the technology used to setup and maintain the communication channel, but just say that all involved parties can just send whatever message they want and it will arrive and be used by the appropriate receiver. However, as described within the design goals in Section 1.2, the network must not be usable at all times, which implies the need to perform offline computation and try to minimize the overall interaction within the selected and designed protocols. Furthermore, as the throughput is limited and might actually be rather low — considering resource-constraint devices that probably communicate via technologies like Bluetooth, NFC or use similar low-power communication channels — we must take care in reducing or preventing unnecessary transmission overheads.

**Communication** Right on top of that generic network infrastructure reside protocols and libraries allowing end-to-end communication, provide structures for data serialization, use standards to ease deployment, enable transparent connectivity to other systems, as well as provide confidentiality, integrity and authenticity on the (lower) layers of the Open Systems Interconnection (OSI) model.

**Key Management** The use of cryptographic primitives, *e.g.* as mentioned to provide link and end-to-end confidentiality, requires the exchange of keying material. This exchange, the choice of Key Derivation Functions (KDFs) and the actual use of appropriate algorithms to achieve the necessary properties is also transparent. As such, it doesn't matter how the keys were exchanged, where they are stored and how they are used, as long as it fits the higher layer usage and is not defined

otherwise at the appropriate sections. We just assume this important part of the overall construction to just work the way we want it. This includes that public keys are surely belonging to the appropriate parties and possibly out-of-band communication happens to achieve the desired goals.

**Storage** Likewise it is assumed that sufficient but finite memory, computational power and communication bandwidth is available when needed. Further the permanent storage-only scenario can be made secure in terms of confidentiality and integrity by usage of appropriate techniques. Several tools at different hierarchy levels, services and devices — *e.g.* disk, partition, file system, file, service — are available for this purpose[2][3][4][5][6].

**Anonymity** Protecting the identity of participating parties is not part of this line of work. Accordingly, techniques focusing on anonymization, pseudonymization, unlinkability or accountability are not discussed. If the proposed schemes should be used within a context that requires such properties, most probably existing techniques for identity protection can be attached to and thus extend the proposed protocols without much interference.

**Malicious Adversary** In general, as defined in the overall design goals in Section 1.2, the aim is to build practical solutions, which are therefore efficient with respect to time and space requirements. This goal is supported by a proper choice of the adversary model. The often used and in Section 1.5.1 specified HBC adversary model is the basis for the following constructions. In case it is unlikely that the adversary behaves within the HBC bounds and his actions are also not trivially encountered by the construction itself, it may be necessary to switch to a stronger adversary model. However, in case such a transition from the HBC model to the malicious model is necessary, it is explicitly stated and explained.

[2] EncFS: `https://vgough.github.io/encfs/` (last accessed: 2015-03-09)
[3] dm-crypt: `https://code.google.com/p/cryptsetup/wiki/DMCrypt` (last accessed: 2015-03-09)
[4] SpiderOak: `https://spideroak.com/` (last accessed: 2015-03-09)
[5] Boxcryptor: `https://www.boxcryptor.com/` (last accessed: 2015-03-09)
[6] GnuPG: `https://www.gnupg.org/` (last accessed: 2015-03-09)

# 2. Related Work

The research field of privacy-preserving comparisons and private searches was quite active over the last years, with many publications addressing very specific requirements. Applications can be found in various domains like record-linkage [HF11; BS10; VCV12], genomics [De 14; JKS08; Ayd+13; Wan+09a] and generic database search [WC09; Cas+14; Ibr+13]. This is a separation of related contributions into use-cases. However, we are more interested in the core functionality that these schemes solve and the extended properties following the constructions. There are two main classes that we distinguish. The first one is concerned with offering a privacy-preserving comparison of elements, while the second class describes secure search schemes to search a remote database in a privacy-preserving way. For the rest of this chapter, our main separation of related contributions follows these two classes. Consequently, we first review schemes to calculate or approximate a distance between two elements, held by different parties. This includes string matching algorithms [BS10; RS10; BK13] or in general data matching [Sca+07; CT12].

The task to perform a secure search by constructing a secure index over a data set for efficient, privacy-preserving document selection [Cas+13; Cas+14] is of interest in many related publications and important contributions to this field are reviewed. In particular, index generation and querying can be done by the same party, which spawns the problem of outsourcing own data and the associated index to an untrusted third party. In the other case, when index generation and searching is performed by different parties, other assumptions must be made which typically result in different methods being used. As both topics — metric and index evaluation — are closely related to each other and also to the constructions within upcoming chapters, a review of the relevant related work for each of these topics will follow.

In Section 2.1 we will review and structure literature related to privacy-preserving metric evaluation, which touches several related fields like string matching, set similarity estimation and privacy-preserving record linkage. Section 2.2 will then look into literature related to secure index evaluation and private search.

## 2.1. Privacy-Preserving Metric Evaluation

Many different metrics were turned into privacy-preserving equivalents or approximations. These schemes themselves often use basic measures like intersection and union cardinality over sets, upon which well known similarity measures like Jaccard similarity coefficient [SKS09] and Sørensen-Dice index [VCV12] are constructed. Next to evaluating set similarity measures, many applications, especially in the document search and genomics domain, require the assessment of similarity between strings. In simple cases such a measure could be the Hamming distance for strings of equal length, or Levenshtein distance, while more complex scenarios could require optimal local or global alignments using Smith-Waterman or the Needleman-Wunsch algorithm. Some of these can by itself be efficiently approximated using set similarity based techniques. Another important domain for metric spaces is the $n$-dimensional space of points, which we group together with measures for $n$-dimensional vectors. All the typical distances like Manhattan, Euclidean, Chebyshev and their generalization, the Minkowski distance belong to this category. The Minkowski distance is also called $l_p$-distance, as it is derived form the $l_p$-norm given by

$$\|x\|_p = \left(\sum_{i=1}^{k} |x_i|^p\right)^{1/p} \qquad \text{for } x \in \mathbb{R}^k, 1 \leq p \leq \infty;$$

Visualizations for $p = 1, 2, \infty$ and $n = 2$ are shown as unit balls in Figure 2.1. In case $n \geq 2$ a point can be interpreted as a vector and between two vectors the cosine similarity is also often used [BMS07].

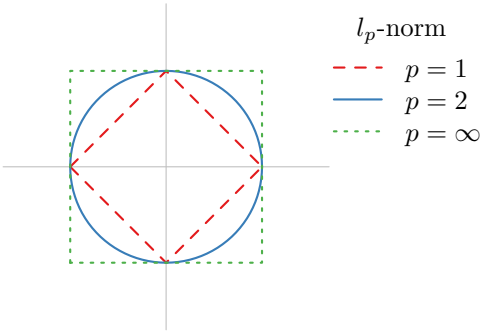



Figure 2.1.: Plot of unit balls for typical $l_p$-norms

There are of course further function domains for which metric spaces exist, examples of measures are given in parantheses:

- Numerical (Manhattan, Euclidean, Canberra, Cosine)

    - Polynomials (Tree or vector measures can be applied)
    - Matrices
- Boolean (Hamming)
- Sets (Jaccard, Dice)
- Strings (Edit Distance)
    - Distance (Damerau-Levenshtein)
    - Matching (Smith-Waterman, Needleman-Wunsch)
- Graphs (Shortest path)
    - Trees (Tree Edit Distance)
    - Bipartite (Hopcroft-Karp)
- Statistical (f-divergence, Kolmogorov-Smirnov statistic, Kullback-Leibler divergence, Wasserstein metric)
    - Distributions (Energy distance, Signal-To-Noise ratio)
    - Dependence (Distance correlation)
- Images (Distance transform, Color distance)
- Software (Cohesion, Coupling, DSQI [Pun09])
    - Testing (Code coverage)
    - Complexity (Cyclomatic complexity)

There are many metrics that can be used to build a comparison scheme and it is not possible to focus on all of them. As the overall use case is to compare two private inputs, might it be documents or genomic sequences, further considerations therefore focus on set and string similarity measures and their privacy-preserving counterparts. However, the presented techniques within this document can also be applied to build privacy-preserving protocols for other similarity measures, under the premise that a fitting embedding can be found with a low distortion and expansion, as described in Section 1.5.

### 2.1.1. Set Similarity

As the evaluation or even estimation of set similarities serves for many applications from biology [Jac01] over computer science [Niw+13] to psychology [Mel+09]. Privacy requirements in all these areas lead to investigations into the private set operation research field. Evfimievski, Gehrke, and Srikant [EGS03] were one of the first to propose a secure protocol for calculating the intersection of two sets between two parties. Such privacy-preserving equivalences of set operations are used as basic building blocks for more complex privacy-preserving systems.

Agrawal and Srikant [AS00] describe how to perform privacy-preserving data mining using set operations, while Cristofaro and Tsudik [CT10] describe usage

examples for a government agency that wants to checking new employees against a criminal record database, Homeland security might be interested in checking a passenger list from a foreign airline against a terrorist database, national law enforcement bodies might want to compare their lists of criminal suspects, or tax authorities might want to learn if suspected tax evaders have accounts on certain foreign banks.

Next to the usage within national agencies, several personalized services exist, which apply profile matching between users. These scenarios include dating sites that match attributes or behavior [Zoo14], deciding whom to share a taxi [WEE14] or a ride with [Sid15]. The information given to such services is mostly regarded as private. Private set operations can in these instances also be a part of the solution to offer equivalent services without having the operator as trusted party gathering all the confidential information. Zhang, Li, and Liu [ZLL13] present such a privacy-preserving protocol to match different user profiles in the context of a social network.

One example for a set similarity measure itself is the Jaccard index [SKS09; BCG11], but many others exist. Jaccard is however often used and also estimated using different privacy-preserving techniques. Singh, Krishna, and Saxena [SKS09] use a semi-honest third party, while Blundo, Cristofaro, and Gasti [BCG11] use Private Set Intersection Cardinality (PSI-CA) techniques to securely estimate the Jaccard index.

To calculate or estimate the similarity of sets, primitive set operations are used and must therefore be supported by the cryptographic primitives involved. Constructions allowing these operations to be performed in a privacy-preserving way are termed Private Set Intersection (PSI) or Private Set Union (PSU).

A combination of set operations on four spheres or sets is depicted in Figure 2.2 for illustration.

Private set similarity systems may use and thus build upon protocols for efficient — linear time complexity in the largest set size — PSI-CA, which works just like PSI protocols, with the limitation of outputting only the final set cardinality. These PSI and PSI-CA [FNP04; KS05; CZ09; SS09] protocols can itself be used as basic privacy-preserving similarity measures. Variations supporting set union [CGT11] are similarly available. Further deviations include PSI with data transfer, which associates arbitrary data — *e.g.* the medical record of a patient — with each set element and finally transfers this data for the elements in the intersection. Authorized-PSI needs client input to be signed by a mutually trusted third party (Certificate Authority (CA)) and in combination PSI-CA can be used to decide whether or not to engage in the actual PSI protocol. The latter variations can be used to limit inference attacks on remote private sets.

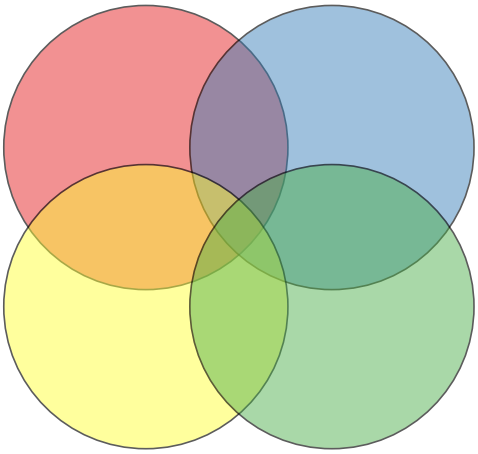


Figure 2.2.: Euler diagram of four circles arranged as shown, each circle representing a set. This is not a Venn diagram, as the sole intersection between the blue and yellow set, as well as between the red and green one is missing. Source: [Mar12]

Some schemes are build to be collusion resistant in the multi-user setting, strengthening the typical Honest-But-Curious (HBC) security model when more than one user is involved [Ker12].

Several methods for constructing PSI schemes are used in the relevant literature. The following list describes some of them.

- Blind signatures [Cha83]
  - Cristofaro and Tsudik [CT10] present PSI schemes and variations like PSI with data transfer, authenticated PSI and variants secure in the malicious model are described.
- Unpredictable Function (UPF)
  - Jarecki and Liu [JL10] use UPFs to build an adaptive PSI in the Random Oracle Model (ROM). This scheme allows the intersection to be calculated interactively, querying for a single entry element at a time and being able to adapt the next query depending on previous results. This is closely related to adaptive attackers and adaptive Oblivious Tranfer (OT). They further allow dummy elements in the sets and queries. The overall scheme is secure against malicious adversaries.
- Oblivious Pseudo-Random Function (OPRF) [Fre+05]
  - Freedman et al. [Fre+05] use OT, Oblivious Polynomial Evaluation (OPE), techniques from single-server Private Information Retrieval (PIR) and symmetrically PIR to build OPRFs and upon these a private keyword search scheme. This search scheme is taken as a special case of set

intersection $X \bigcap Y$, where one set $X$ consists of the actually searched keyword and thus has cardinality $|X| = 1$. A set intersection protocol is then constructed by applying the keyword search scheme for every element in $X$.
    - Hazay and Lindell [HL08] describe a generic PSI protocol secure against malicious and covert adversaries. Covert adversaries lie between HBC and malicious ones, as they try to cheat, but also try to not be detected. They will rather not cheat than being detected. See Section 1.5.1.
- Oblivious Polynomial Evaluation (OPE) [NP99]
    - Freedman, Nissim, and Pinkas [FNP04] propose schemes for two-party set intersection private matching secure in the HBC and malicious ROM. Variations allow the computation of the intersection cardinality, other functions of the intersection, approximation of the final set size, fuzzy matching and an extension to multi-set intersection.
    - Hazay and Nissim [HN10] extend the work of [FNP04] to the malicious attacker model, also a variation secure in the standard model — opposed to the often used ROM — is described. Together with these security gains follows a large overhead through using many invocations of Zero Knowledge Proofs (ZKPs) and OTs.
    - Kissner and Song [KS05] also describe PSI secure in the standard model against malicious attackers, however through the use of generic, not specialized ZKPs, overheads are quite large. They also additionally achieve mutual PSI and allow more than two parties or sets to be operated on. Further operations can easily be combined to generate only the desired final result.
    - Dachman-Soled et al. [Dac+09] improve upon [KS05] in replacing the generic ZKPs with Shamir's secret sharing techniques to gain efficiency.

On top of set similarity measures together with their privacy-preserving equivalents, privacy-preserving data analyses schemes can be constructed [Dom08].

### Locality-Sensitive Hashing

A more efficient and also less private way of matching elements is to use Locality-Sensitive Hashing (LSH). These hash algorithms describe a technique to map similar elements to identical values, allowing plaintext identity checks to quickly estimate the similarity of sets. Elements from two sets are hashed into bins, if there is a large overlap in the used and unused bins for both sets, then the sets are declared similar. The properties of the herein described hash functions are contradictory to the definition of cryptographic hash functions. Figure 2.3 depicts this difference.

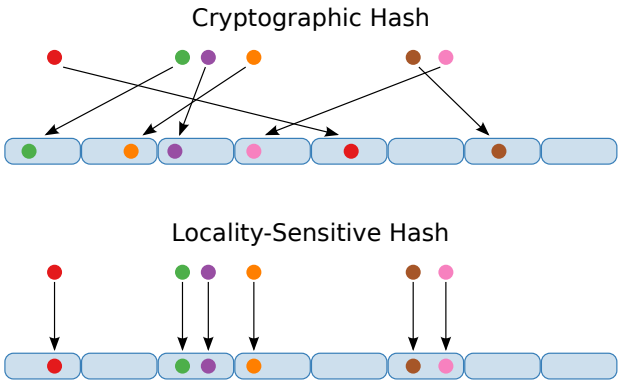



Figure 2.3.: Visualization of hashing elements into bins using a cryptographic hash and a locality-sensitive hash. The multicolored circles are elements from the domain of the hash function, while the blue rectangles with rounded corners are elements from the co-domain of the hash function. Elements drawn closely together are more similar than those drawn farther away from each other.

LSH as introduced by Indyk and Motwani [IM98] describes a set of techniques for generating hashes given a certain input. Cryptographic hash functions should exhibit the avalanche effect property, which states that upon inversion of a single input bit, on average 50% of the output bits are also inverted. Further, the strict avalanche criterion even requires that each output bit has a probability of 0.5 to be changed upon the inversion of any single input bit. This desired property helps in achieving the concept of *diffusion*, as stated by Shannon [Sha49]. These are, next to the traditional goals of cryptographic hash functions (pre-image, second pre-image and collision resistance) important properties to approximate the concept of a random function, *e.g.* as used in the Random Oracle Model (ROM). Such a random function approximation requires that an adversary also learns no useful information about the input given the resulting hash.

LSH functions are grouped into families, which are bound to a certain metric space. Such a family $\mathcal{F}$ for a metric space $(A, d)$ provides functions, *i.e.* $f$, such that close values $x, y \in A$ map to the same hash value with high probability $P[f(x) = f(y)] \geq p$. Indyk and Motwani [IM98] initially define LSH schemes. Given a distance measure $d\colon S \times S \to \mathbb{R}$ (not necessarily a distance metric), let $B(q, r) = \{p : d(q, p) \leq r\} \quad p, q \in S, r \in \mathbb{R}$ be a ball of elements around and thus similar to $q$ by the distance measure $d$. Similarity is defined as the distance $d$ being lower or equal to $r$. The ball $B(q, r)$ therefore includes all elements around $q$ within distance $r$. A family $\mathcal{H} = \{h\colon S \to U\}$ is called $(r_1, r_2, p_1, p_2)$-sensitive for $d$ if for any $q, p, p' \in S$

$$\text{if } p \in B(q, r_1) \text{ then } P_{\mathcal{H}}[h(q) = h(p)] \geq p_1,$$
$$\text{if } p' \notin B(q, r_2) \text{ then } P_{\mathcal{H}}[h(q) = h(p)] \leq p_2.$$

For a family to be interesting the inequalities $r_1 < r_2$ and $p_1 > p_2$ must be satisfied.

**AND & OR Constructions** Several independently chosen hash functions $h_1, h_2, \ldots, h_m \in \mathcal{H}$ from the same $(r_1, r_2, p_1, p_2)$-sensitive family can be combined to form a new family $\mathcal{H}'$. Two generic ways to build hash functions in $\mathcal{H}'$ are often used [LRU14]. The *AND-construction* specifies that for a fixed number of $m$ independently chosen hash functions from $\mathcal{H}$, as defined above, a function $h' \in \mathcal{H}'$ evaluates to the same value for two different elements $h'(p) = h'(q)$ if and only if $h_i(p) = h_i(q) \quad \forall i \in [1, m]$ all hash functions from $\mathcal{H}$ also evaluate to the same value. The generated family $\mathcal{H}'$ is $(r_1, r_2, p_1^m, p_2^m)$-sensitive. Intuitively then an *OR-construction* is defined as taking independently a fixed number of $n$ hash functions from the family $\mathcal{H}$ to build a family $\mathcal{H}''$, in which a function $h'' \in \mathcal{H}''$ evaluates to the same value $h''(p) = h''(q)$ if $h_i(p) = h_i(q) \exists i \in [1, n]$. The generated family $\mathcal{H}''$ is $(r_1, r_2, 1-(1-p_1)^n, 1-(1-p_2)^n)$-sensitive. This way also combinations of AND- and OR-constructions can be build to amplify an LSH family in its sensitivity.

MinHash [Bro97] and variants are regularly used to estimate the Jaccard similarity. Leskovec, Rajaraman, and Ullman [LRU14] give LSH constructions for the Hamming distance, for the cosine distance using random hyperplanes and for the Euclidean distance.

Such LSH instantiations can still be used as input to set intersection and union protocols. LSH also preserves some level of privacy, which was evaluated for different LSH parameters by Aghasaryan et al. [Agh+14].

There is also quite some research using similarity measures in the traditional way, that is, upon privacy-preserving building blocks to infer some sort of similarity [SBR09]. This area is not explored further, as the main interest is in evaluating or estimating a similarity measure effectively, efficiently and privately, without trading privacy guarantees against necessary trust in additional parties.

### 2.1.2. String Matching

String matching specializes privacy-preserving metric evaluation by explicitly restricting the metric domain to strings over a certain alphabet $\mathcal{A}$.

Applications applying string matching one way or another are rich and can be found everywhere:

- *Search*: Search for or within a document using a term (web search, file search, searching an index)

- *Compare*: Find duplicates given a set of words or documents (matching database entries, finding plagiarism)
- *Test*: Extract properties about the string source (Check for characteristic sub-strings for classification)
- *Construct*: Building larger sequences using string overlaps (genome sequencing and overlapping readouts)

Research into string matching algorithms is defined by a long list of proposed algorithms over many years and for many different problems. String matching itself is closely related to the distance between strings, which can be measured by a large variety of means, ranging from generic and simple solutions like the Hamming distance [Ham50] to more powerful algorithms like Smith-Waterman [SW81] solving local sequence alignment problems. A survey about current developments can be found in [LH10].

As several tasks, for example checking whether a user profile is within a remote database, do not require the exact distance between two strings, data items or other entities, the notion of approximate matching was introduced to define levels of similarity, which in the most extreme way only output a single bit of information: if the input strings are similar or not. Due to these properties this class is called approximate string matching algorithms, which is not to be confused with the approximate string matching of [HD80], where the term "approximate" referred to the property of two strings being close in distance.

Two of the applications for string comparison algorithms which are often used for motivation are calculating the distance of genome or protein sequences in the life sciences and checking if a person is present in a remote database. As these topics by design deal with very personal information, which must not be given to third parties, the necessity to build privacy-preserving matching algorithms arose. As these were not sufficient to protect privacy due to information leakage given by the exact distance results, just obtained in a privacy-preserving manner, combinations of the above mentioned approximation and the privacy-preserving computational steps were developed. A survey of recently published algorithms together with benchmark results can be found in [BS10].

Estimating or evaluating the similarity between strings in a privacy-preserving way can be done with many different techniques. We will now investigate selected literature for relevant constructions. As such we will first review constructions based upon a Trusted Third Party (TTP), the next section then describes proposals that map strings to sets and estimate string similarity using privacy-preserving set similarity measures. Following this, we will introduce proposals based on the secure evaluation of the original distance or similarity measure and finally comes a section which presents schemes with further enhanced functionality based on the oblivious evaluation of a finite state machine.

### Trusted Third Party (TTP)

To solve the privacy issues and fulfill derived requirements, one can introduce a party that is fully trusted by all participants, which input data. Such a TTP would receive all required (sensitive) information, evaluate the desired functionality and return correct results to the appropriate parties. A contract between the TTP and inputting participants legally binds the TTP not to misuse the information. Further the actions of the TTP could to some degree be monitored and controlled. However, all participants must still trust the TTP not to misuse the data, as this is still technically possible.

The risk and required trust in the TTP can be reduced by reducing the TTP calculations to the absolutely necessary minimum. As much as possible would be computed locally. The transferred data must then also be reduced to the absolutely necessary information that the TTP needs to carry out the desired task. Further more, data and computations can be split across several TTPs, which are unlikely to collude.

String matching using a TTP was, for example, proposed by Schnell, Bachteler, and Reiher [SBR09] and Durham et al. [Dur+12]. Their constructions are highly efficient when compared to other privacy-preserving string matching solutions and can evaluate complex functionality more easily than using cryptographic primitives. This leads to solutions which support approximate matching, use only a minimum of interactivity between the parties and can even be size-hiding in the input. Further, clients inputting data experience a very low overhead do to absence of cryptography for input confidentiality. A short evaluation regarding our requirements is depicted in Figure 2.4. The check mark (✓) symbolizes that a requirement is fulfilled, equivalently an X (✗) shows that it is not fulfilled and a combination of both (✓/✗) stands for partial fulfillment.

| Priv. | Effi. | Appr. | Non-Interac. | Two-Party | Thin Client | Hide Size | Infer. |
|---|---|---|---|---|---|---|---|
| ✓/✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |

Figure 2.4.: Fulfillment of requirements by selected Trusted Third Party systems. [SBR09; Dur+12]

Many use cases require higher privacy guarantees than can be given in the TTP scenario, also a TTP setup is not fitting our desired two-party construction. Inference control would be feasible to be performed by the TTP, is however not discussed for such schemes in the literature.

### Set similarity

As presented in Section 2.1.1, string matching can be performed upon private set operations. Normally strings are converted into sets and these sets are compared with privacy-preserving set similarity measures.

One of the protocols introduced by Schnell, Bachteler, and Reiher [SBR09] uses Bloom filters to represent strings and transforms the notion of distances between strings into distances between similar Bloom filters. We will also use Bloom filters as set representation for our strings and build the matching protocol upon them. However, we use a two-party technique for comparing the Bloom filters and therefore do not need a trusted third party for comparing the strings. Furthermore, our protocol can be size-hiding, by choosing appropriate Bloom filter sizes, that are not proportional to the string length. Bloom filters by itself were also subject to further privacy enhancements by inserting random bits, injecting fake elements [KVC12], using keyed hashes [BC04], interleaving several elements within a single Bloom filter [SBR11] or composing several Bloom filters [Dur+14].

However, relying on the obfuscation property of Bloom filters as sole privacy protection may not deliver desired privacy guarantees. To judge the possible privacy gains by using Bloom filters, Bianchi, Bracciale, and Loreti [BBL12] analyzed the privacy achieved using normal Bloom filters. Further, Kuzu et al. [Kuz+11] showed that Bloom filter encodings with typically used parameters — encoding Surnames, usage of 15 hash functions and a Bloom filter length of 500 bit — can lead to large data leakage (11% of the data was discovered) The collected data was processed and represented as a constraint satisfaction problem (CSP), which was solved by their CSP solver. Niedermeyer et al. [Nie+14] used similar parameters for the filter, but a different technique based on frequencies of certain substring lengths. They discover 12% of the encoded data. Kuzu et al. [Kuz+12] proposed to use CSP for multiple identifiers, but did not find successful frequency attacks using their solver. Kroll and Steinmetzer [KS15] developed an automated tool for discovering forenames, surnames and place of birth.

| Priv. | Effi. | Appr. | Non-Interac. | Two-Party | Thin Client | Hide Size | Infer. |
|---|---|---|---|---|---|---|---|
| ✓ | ✓ | ✗ | ✗ | ✓ | ✓/✗ | ✓/✗ | ✗ |

Figure 2.5.: Fulfillment of requirements by generic Secure Multi-Party Computation Private Set Intersection systems. [HEK12]

Alternatively, techniques from Private Set Intersection (PSI) could be used. Huang, Evans, and Katz [HEK12] use garbled circuits to implement PSI algorithms and compare them to algorithms using homomorphic encryption and other specialized PSI designs, its properties are given in Figure 2.5. Due to the use of garbled circuits, the client has a rather large setup overhead and the protocol is

interactive. The most problematic part however is that the protocol reveals the actual intersection, that is the elements which are in both sets. Inference attacks are fueled when more information than necessary is revealed.

| Priv. | Effi. | Appr. | Non-Interac. | Two-Party | Thin Client | Hide Size | Infer. |
|---|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓/✗ | ✓ | ✓ | ✓ | ✗ | ✗ |

Figure 2.6.: Fulfillment of requirements by selected custom Private Set Intersection Cardinality systems. [CGT11; CT10; Bal+11]

As revealing the content of the intersection is not appropriate for a privacy-preserving protocol that measures string distances, protocol variations with less information in the output are to be found. Based on these security concerns, protocols for Private Set Intersection Cardinality (PSI-CA) were developed [CT10]. As the name suggests, these protocols also perform PSI, but they only output the cardinality of the result, not the intersection itself. Yet, these solutions still reveal the actual cardinality, whereas it might be desirable to only reveal whether there is a match within a certain range. Figure 2.6 shows how these systems match our design requirements. Further [Bal+11] presents a more efficient solution, but which only matches exact strings, whereas we compare approximate strings.

Many different proposals to implement PSI or PSI-CA can be found in literature, as was roughly sketched in Section 2.1.1. Another influence comes from the actual data that is to be matched. If the set elements contain high entropy and are drawn from a large domain, very simple techniques like cryptographic hashing and comparing hashes might be sufficient as shown by Nagy et al. [Nag+13], while other schemes for rather small domains map a set to a polynomial and perform set intersection via operations on them [FNP04; KS05]. The polynomials must have a degree linear in the size of the domain, with polynomial addition being an equivalent to calculating the set intersection as shown by Kissner and Song [KS05]. However, the required multiplications of the polynomial representations are rather expensive with a quadratic computational complexity in the size of the domain.

### Secure Dynamic Programming (SDP)

All solutions presented above only estimate the actual string similarity measure using a low distortion embedding from the string similarity metric space into the set similarity metric space. While results are quite good, given the efficiency and privacy achieved, results are never known to be correct. For highly accurate and correct results, the original similarity functions must be reassembled closely by the privacy-preserving protocol.

Using generic techniques to achieve input-privacy in multi-party computation, that is Secure Multi-Party Computation (SMPC) — *e.g.* through the use of garbled-circuits or homomorphic encryption — any functionality can be evaluated securely. However evaluating complex algorithms this way introduces large overheads. Nevertheless, protocols to securely compute dynamic programming algorithms, that is the Levenshtein distance, Smith-Waterman or Needleman-Wunsch were presented [JKS08; AKD03; RS10]. Figure 2.7 depicts how the properties of evaluating the original dynamic programming algorithm through secure computation matches the requirements given in Section 1.2.

| Priv. | Effi. | Appr. | Non-Interac. | Two-Party | Thin Client | Hide Size | Infer. |
|---|---|---|---|---|---|---|---|
| ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |

Figure 2.7.: Fulfillment of requirements by selected exact Secure Dynamic Programming systems. [JKS08; AKD03; RS10]

### Oblivious Finite State Machine (OFSM)

Troncoso-Pastoriza, Katzenbeisser, and Celik [TKC07] presented a protocol that obliviously evaluates a Finite State Machine (FSM) between two parties. The client builds the FSM such that it accepts a string up to a certain edit distance away from its own input and then jointly executes the FSM together with the server. The server obliviously inputs a single character at a time. After the whole server input is used, the FSM is checked if it reached an end state. This allows rich functionality, exact and still privacy-preserving results with some sort of inference control. The efficiency of this solution however is far from acceptable. Figure 2.8 matches the proposal to our requirements.

| Priv. | Effi. | Appr. | Non-Interac. | Two-Party | Thin Client | Hide Size | Infer. |
|---|---|---|---|---|---|---|---|
| ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓/✗ |

Figure 2.8.: Fulfillment of requirements by selected exact Oblivious Finite State Machine system. [TKC07]

### String Matching Summary

Different basic building blocks were presented to construct privacy-preserving string matching systems. Relevant literature using these building blocks was reviewed and the respecting properties were matched against our requirements. Table 2.1 combines all previously seen figures for fulfillment of our requirements. Furthermore, Oblivious RAM (ORAM) is added wich perfectly hides access patterns but

| Scheme | Priv. | Effi. | Appr. | Non-Int. | 2-Party | Thin | Size | Infer. |
|---|---|---|---|---|---|---|---|---|
| TTP [SBR09; Dur+12] | ✓/✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| PSI [HEK12] | ✓ | ✓ | ✗ | ✗ | ✓ | ✓/✗ | ✓/✗ | ✗ |
| PSI-CA [CGT11; CT10; Bal+11] | ✓ | ✓ | ✓/✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| SDP [JKS08; AKD03; RS10] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| OFSM [TKC07] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓/✗ |
| ORAM [Gol87] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| Our 4,5,6 [BK13; KBS14; Bec15] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 2.1.: Fulfillment of requirements by selected privacy-preserving string matching protocols.

comes at a very high cost. The overall scheme presented in this thesis is also depicted in the last row.

Further related is literature on outsourcing string comparisons. Two parties have private input and want a third party to evaluate a function over it. This is a typical use case for two or more similar clients and is often used in the Cloud Computing scenario. Next to generic outsourced algebraic operations [BA08; Kha05; Wan+11] also the special case of outsourcing string comparisons or set operations [AL05; Bla+12a; Ker12] is studied widely. Outsourcing specifically the matching of genomes is considered by Chen et al. [Che+12], who split the task in coarse- and fine-granular matching, with the coarse and computationally intense matching being outsourced to public Cloud compute providers, while the detailed matching for selected parts is done on a private, trusted Cloud. The discussed scenario is read mapping, which is the matching of a private genome against a publicly available reference genome. Kantarcioglu et al. [Kan+08] presented how simple queries can be performed on outsourced genomic data.

### 2.1.3. Numerical distances

As posed in list 2.1, next to the similarity between sets or strings, many other distance measures exist. Other important, often used and for our purposes closely related measures are those termed "numerical" in the aforementioned list. This groups together distances between points, polynomials and similar mathematical structures.

The problem of calculating the Manhattan and Euclidean distance for Private Similarity Search (PSS) was considered by Du and Atallah [DA01] and Laur and Lipmaa [LL04]. They build protocols for private similarity search systems — very similar to the problem in this thesis — in which a querier Alice and a database owner Bob interact with each other. Alice inputs a vector for which Bob should find the closest vector in his database using the $l_1$- or $l_2$-distance. The proposals use a TTP to restrain from employing much less efficient secure computation that could be based on generic secret sharing or customized protocols.

## 2.2. Searchable Encryption

We will now leave the area of one-to-one comparisons of elements, but rather look at secure indexes, which are compared to single queries for finding elements that are similar. The use case most often is outsourcing of own data and later performing a search over it. However, this area is still closely related to our overall topic.

In cases when usage of remote storage is unavoidable (may it be due to limited local resources) and the storage provider does not enjoy the users unconditional trust, outsourced data must be protected. Most importantly the main protection goals of confidentiality and integrity of stored data must be ensured by the use of appropriate cryptographic primitives. These requirements normally translate into encrypting and appending authentication codes onto the data. However, for any kind of utilization, data must be retrieved and decrypted. While there are now generic Fully Homomorphic Encryption (FHE) schemes to operate even arbitrarily upon encrypted data [Gen09], they are very computationally, communicationally and memory-intense. Furthermore, these generic secure computation schemes offer potentially richer functionality than necessary. One kind of rather simple but often used functionality is search.

A typical use case might look like this: User $A$ wants to outsource storage of own data to an external, not fully trusted party. The user still wants to perform some kind of search over an previously generated index without downloading the whole index and performing the search operation locally. The quest for constructing such a secure and efficient index scheme was first solved by Song, Wagner, and Perrig [SWP00]. Du and Atallah [DA01] classify four models in which different settings for searching over a database or document collection require the query to be kept private.

These models are depicted in Figure 2.9, where Bob can always be called the server $S$. Model (a) Private Information Matching (PIM) denotes the two party setting, in which Bob (denoted by $S$) has his own private database, which should be searched by Alice (denoted by $A$). It is required to keep the database as well as the query private, so the trapdoor, as well as the resulting answer to Alice must leak as few information as possible. In case Bob does not serve a private, but a public database — depicted as (b) Private Information Matching from Public Database (PIMPD) model — the whole setting transforms slightly, as replies may leak additional information about the public database, while query privacy must still be preserved. Both models share the property, that the server $S$ most probably knows the content of the database and thus, if trapdoors are matched directly and not through some Secure Computation (SC) scheme, learns information about the query more easily.

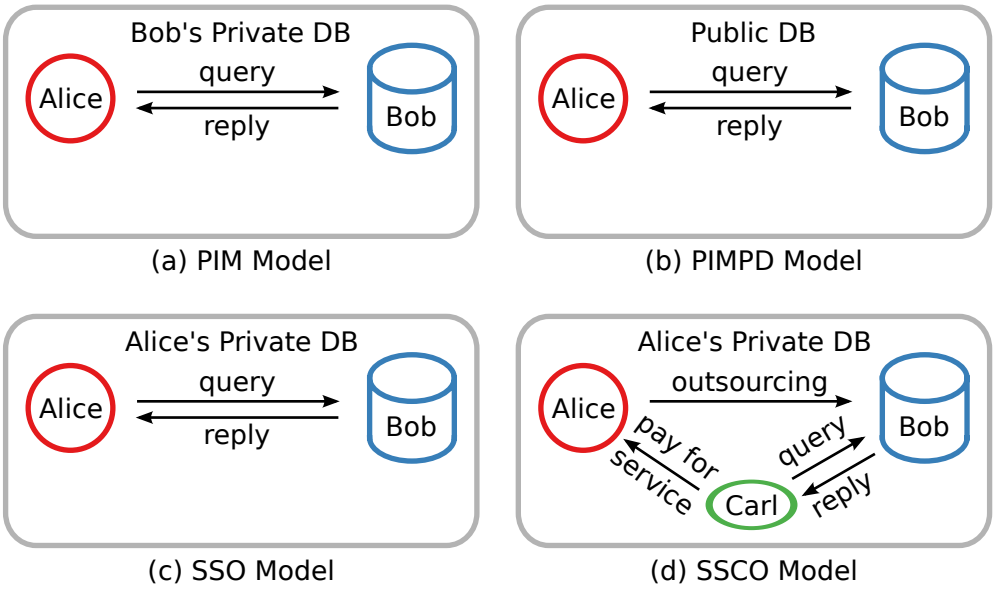



Figure 2.9.: Models for privately querying a remote database. Source: Du and Atallah [DA01]

Contrary to the first two models, where there are two parties providing input to the search scheme, model (c) Secure Storage Outsourcing (SSO) has only Alice as database and search query provider. The server $S$ only serves as a HBC external storage and compute provider that enables outsourcing of storage. The last model (d) Secure Storage and Computing Outsourcing (SSCO) even introduces a third user $C$, which queries the outsourced database of $A$ on Server $S$. The SSO model is typically considered for Symmetric Searchable Encryption (SSE) schemes, as described in further detail in Section 2.2.1, whereas the SSCO model fits the Public key Encryption with Keyword Search (PEKS) schemes, described in Section 2.2.2.

A trapdoor or token, constructed from a query, is typically used to perform the search on the server side. Searchable encryption is most often used as secure (reverse-) index, which maps a word $w$ to a set of documents, often called document collection $\mathcal{D}$, which are tagged with the word $w$. Upon evaluation of a queried word, most schemes return the set of document indexes $\mathcal{D}(w)$, which contain or are tagged with the word $w$. The client can then decrypt and use the retrieved indexes on its own discretion.

Such secure indexes find different applications. An often used motivation is to perform a search over encrypted data outsourced in the Cloud Computing paradigm while storing and querying confidential information like government documents, hospital records, personal emails, sensitive business data or audit

logs [Goh04]. Search schemes can therefore be found in many variations, some key contributions came from Dan Boneh et al. [Dan+04], Chang and Mitzenmacher [CM05], and Song, Wagner, and Perrig [SWP00]. Next to the solutions for search over private data are also proposals for private accumulated hashing [BM93; Nyb93] as solved in [Goh04] and subset or range queries [BW07; Goh04].

The first Searchable Encryption (SE) constructions used security definitions known from symmetric, deterministic encryption systems, notably indistinguishability of ciphertext from truly random data [SWP00]. It was then shown that the encrypted index is indeed computationally indistinguishable to random data. However, an adversary can and will in practice not restrict itself at looking at the encrypted index, but also use the supplied trapdoors to learn information about queries and the index. These ciphertext-only (or index-only) attacks were therefore not strong enough to withstand adversaries which took search queries and results into account.

Goh [Goh04] defines semantic security against adaptive chosen keyword attack (IND-CKA)using a game-based security definition and providing semantic security in the SE model. The game is played as follows. The challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ two documents $V_0, V_1$ of equal length but possibly different numbers of contained words. $\mathcal{C}$ selects a bit $b$ uniform at random and gives $\mathcal{A}$ the encrypted index for $V_b$. The challenge for $\mathcal{A}$ is to decide from which document $V_0$ or $V_1$ the index was created. If deciding this problem is hard, then it must also be hard for any (key-) word used inside the index. Chang and Mitzenmacher [CM05] propose an even stronger simulation-based definition that is called (IND2-CKA) by Goh [Goh04]. It reduces the assumptions over Semantic Security Against Adaptive Chosen Keyword Attack (IND-CKA), as it additionally allows the documents $V_0$ and $V_1$ to have unequal length. Curtmola et al. [Cur+06] show that all previous schemes and security definitions only included non-adaptive adversaries. That is, attackers, who do not include results from previous queries as knowledge to influence future queries. Further, all previous schemes are just secure against an adaptive adversary, when all queries for the system are made at once. Indistinguishability using simulation-based definitions for the adaptive setting are given.

Searchable encryption schemes are typically described by a quadruple of polynomial time functions (`Keygen`,`Trapdoor`,`BuildIndex`,`SearchIndex`). The following definition from Curtmola et al. [Cur+06] describes their intention and usage.

- `Keygen`$(1^k)$ is a probabilistic key generation algorithm that is run by the user to setup the scheme. It takes a security parameter $k$ and returns a secret key $K$ such that the length of $K$ is polynomially bound in $k$.
- `Trapdoor`$(K, w)$ is run by the user to generate a trapdoor for a given word. It takes a secret key $K$ and a word $w$ as inputs, and returns a trapdoor $T_w$.

- `BuildIndex`$(K, \mathcal{D})$ is a (possibly probabilistic) algorithm run by the user to generate indexes. It takes a secret key $K$ and a document collection, of size polynomially bounded in $k$, as inputs and returns an index $\mathcal{I}$ such that the length of $\mathcal{I}$ is polynomially bounded in $k$.
- `SearchIndex`$(\mathcal{I}, T_w)$ is run by the server $S$ in order to search for the documents in $\mathcal{D}$ that contain word $w$. It takes an index $\mathcal{I}$ for a collection $\mathcal{D}$ and a trapdoor $T_w$ for word $w$ as inputs and returns $\mathcal{D}(w)$, the set of identifiers of documents containing $w$.

The proposed SE schemes can offer additional functionality or further desired properties next to actually querying a secure index. The most prominent properties are listed below and are briefly introduced.

- *Private-key* setting: User encrypts and outsources own data to external party. Can achieve sub-linear search time in the index-size, which implies loosing semantic security. See Section 2.2.1.
- *Public-key* setting: Third party encrypts data and stores it on a searchable server. Different users can search and retrieve data. Typically build on homomorphic encryption or functional encryption schemes. Can achieve semantic security for the query under adaptive chosen-keyword attack and has optimum time complexity sub-linear in the index size. See Section 2.2.2.
- *Keyword* search: There is a predefined set of keywords on index construction time, which can be searched for. These schemes do not naturally support search over different hierarchical levels, *e.g.* if the system should allow keyword search over a natural language text and the keywords used to build the index are words within the language, then it is not trivial to search for a morpheme, phrase, clause or sentence. If the keywords used to construct the index are 3-grams, then a search for 2-grams is in the same way not trivially possible. All schemes use keywords one way or another, as they all rely on a secure index and this index typically stores ⟨Keyword,DocumentID⟩ pairs.
- *Fuzzy keyword* search: Supports matching for word deviations, which are similar to a keyword under some certain similarity measure. [Wan+14; Li+10; BC14; Liu+11; Ibr+13]
- *Dynamic/Updatable*: Supports efficient updating of the secure index, more efficient than reconstructing the whole index for every change on the index or data set [Cas+14; SPS14; NPG14; KP13; KPR12]. These updates may leak information about the updated keywords [KPR12; NPG14], require the client and server to store additional data [Cas+14] or schemes may have an index size in the order of number of documents times number of keywords [KP13].
- *Multi-keyword*: Supports search for multiple keywords at once, which should be more efficient and concealing than performing multiple single keyword searches and returning all results to the searching party for evaluation of operators. Can be split regarding supported logical operators *conjunctive* [Ker11; GSW04; BW07; WWP08], *disjunctive*, *arbitrary* boolean queries [Cas+13] or even regular expressions [Wei13; Sal+14].

- *Ranked*: Supports ranking of results. Results are equipped with a similarity score against the query and can therefore be ranked according to the used similarity measure. [Cao+14; Cao+11]
- *Multi-Key/Multi-User/Delegation*: Allow several different users, having different keys to search the same data (secure index). Can be implemented using a leveled homomorphic cryptosystem and attribute-based encryption. Offers the additional functions AddUser, RevokeUser to the basic set of SE functions. [Cur+06; EOM14]
- *Malicious Server*: Using universal arguments [BG02] and memory checking [Blu+91] a SE scheme can be made secure in the malicious server model. [Wei13]
- *Adaptive Security*: A system secure under the adaptive, simulation-based security model given by Curtmola et al. [Cur+06]. [NPG14; Cur+06; Cas+13; Liu+14]
- *Probabilistic*: The search system returns false-positives or misses true-positives with low probability. [Goh04]

A natural follow-up step after performing a search, would be to either request and retrieve the indexed documents $d_i$, or to select these documents for further remote processing. Both steps can reveal the securely obtained indexes. Therefore a PIR scheme can be used to retrieve the document set without leaking information about the actual indexes. With regards to selection for further processing, a FHE or Leveled Homomorphic Encryption (LHE) system can be used to evaluate the filter function over the database, with input from the client (the homomorphically encrypted indexes) and from the server (the actual documents within the database).

What follows is a detailed review of the literature for private and public key SE schemes.

### 2.2.1. Private Key

Using a private key based scheme for constructing the index and trapdoors maps well to the SSO model presented in [DA01] and Figure 2.9. So typically a client $A$ wants to outsource storage of his confidential data to a third party $B$.

Song, Wagner, and Perrig [SWP00] gave one of the first solutions to the searchable secure storage outsourcing problem. The idea to build the secure index is to convert all data into text-only documents, encrypt every word $W_i$ deterministically using a Pseudo-Random Permutation (PRP) — a block cipher — and further xor'ing $\oplus$ every block cipher encrypted word $E(W_i)$ using a random seed $S_i$ followed by a stream generated from the Pseudo-Random Number Generator (PRNG)

$F_{k_i}(S_i)$, $C_i = E(W_i) \oplus (S_i | F_{k_i}(S_i))$. $k_i$ is generated from $W_i$ using a keyed Pseudo-Random Function (PRF) and | denotes concatenation. The index is a composition of all $C_i$.

This specific structure supports matching of a trapdoor $E(W_i')$. The search then works as follows: The client $A$ has a word $W_i$ he wants to search for in the document. From this he generates the trapdoor $X = E(W_i)$ as well as the stream cipher key $k_i$ and sends $\langle X, k_i \rangle$ to the third party $B$. $B$ goes over all double-encrypted words in the index and generates $(S_i' | F_{k_i}(S_i)') = C_i \oplus E(W_i)$. At last he generates $F_{k_i}(S_i')$ and checks if $F_{k_i}(S_i') = F_{k_i}(S_i)'$. In case the generated stream matches the found one, the search for the token $E(W_i)$ was positive and can return the document and position where the hit occurred. The used techniques fall under the hash-table paradigm, where (key-) words are deterministically mapped to random values and can be found using an appropriate lookup.

The Song, Wagner, and Perrig [SWP00] scheme supports only exact matches, leaks search and therefore access patterns. A search for multiple keywords at once leaks the same amount of information to the server as searching for all keywords individually. A malicious server is not supported, as well as ranked results or multiple users under different keys. However, dynamic updates can be introduced easily. The search time or computational complexity $\mathcal{O}(n)$ is linear in the size of the index $n$. The index was proven to not leak information (other than number and length of documents) as long as it is not queried.

Goh [Goh04] constructs a more efficient scheme, which computational complexity is only linear in the number of documents $m$ indexed $\mathcal{O}(m)$. This is due to a per document Bloom filter (see Section 3.4), which by itself can be checked in constant time $\mathcal{O}(1)$. All keywords for a document are added to the corresponding Bloom filter. All keywords are salted using a secret master key and the document identifier. All Bloom filters are chosen to have the same size, giving space for enough keywords on the largest document. After keyword insertion, random 1s are inserted into all Bloom filters, so that all Bloom filter contain the same amount of inserted keywords and cannot be differentiated by their weights. The search works straight forward. The server receives the key-salted hashes for the keyword, that are hashed again by the server using the document IDs as salt. The final hashes are used as indices to the Bloom filters to check if the respective keyword was inserted into the corresponding document. The scheme is proven IND-CKA secure given their introduced definition of IND-CKA.

Curtmola et al. [Cur+06] describes two efficient SSE constructions, set within the two security classes defined by themselves. Their first scheme "SSE-1" is secure in the non-adaptive setting, whereas their enhanced scheme "SSE-2", which requires higher communication costs and server storage, is also secure in the adaptive attacker model. These schemes are constructed using an array $R$ which contains

entries to linked lists, where each list represents the set of documents linked to a single keyword in the searchable encryption scheme. Next to this array a hash table is used, which contains information to locate the corresponding position for a queried keyword in the array $R$ and to decrypt the linked list that is attached at the array position. The decrypted list gives all the document IDs which link to documents containing the keyword $w$.

Liu et al. [Liu+14] describe and mount a practical attack on search queries in the searchable symmetric encryption setting. As query encryption is deterministic, query frequencies for a keyword can be recorded. This frequency information is matched with auxiliary frequency information about search queries, for example using Google Trends[1], Google n-grams [LMA12] or resources for specific domains, like word frequencies over PubMed[2]. After collecting keyword frequencies, the trapdoors could be linked to the corresponding keywords with high probability. Based on this Liu et al. [Liu+14] propose the straight forwards enhancement of querying several keywords at random together with the proper keyword, further the index can be grouped and a random query is sent to each index group to blur the keyword frequencies even more. The resulting scheme is attributed *group-based*. They show that such randomized requests decreases the accuracy of the proposed attacks significantly. It could be possible to also mount the described attacks on indeterministic encryption schemes as they are used in the public key setting described in Section 2.2.2. The idea is that identical queries will generate the same results or access patterns and in such a way also allow to acquire frequency information about the keywords used for the indeterministic trapdoors.

### 2.2.2. Public Key

Public key searchable encryption — PEKS [Dan+04] — schemes are typically placed in a three party setting, where user $A$ constructs an index and outsources it to a less trusted store and compute server $B$, while a third party $C$ can issue queries against the secure index using the public key of $A$ to generate trapdoors. Such schemes can be proven secure under definitions like semantic security against adaptive chose keyword search [Goh04]. Bellare and Boldyreva [BB07] introduced asymmetric Efficiently Searchable Encryption (ESE), which allows users with access to the public key to extend the index by adding keywords and also generate trapdoors to search the index. The scheme is deterministic and achieves sub-linear search time, which is bought by leaking access patterns.

Dong et al. [Don+13] proposes an interactive public key encryption with fuzzy keyword search (IPEFKS) scheme. They use an embedding from the edit distance

[1]Google Trends: https://www.google.com/trends/ (last accessed: 2015-03-09)
[2]PubMed: http://www.ncbi.nlm.nih.gov/pubmed (last accessed: 2015-03-09)

string metric to the Hamming distance from Ostrovsky and Rabani [OR07] and measure closeness as Hamming distance between a keyword query and the inverted index of the document store using the homomorphic encryption scheme from Fan and Vercauteren [FV12]. For generation or insertion of elements into the encrypted index, pairing based cryptography is used to check of an entry for a certain keyword already exists.

The public key search schemes described above can be generalized under the term Functional Encryption (FE). We will now briefly describe this generalization to have an overview on the respective field and can put the presented SE schemes into line with other enhanced encryption schemes, which come together under the FE umbrella.

FE, introduced by Boneh, Sahai, and Waters [BSW11] describes a combination of several lines of work, which offer additional functionality for encrypted data. In general FE enhances traditional encryption schemes by introducing a functionality $F$ and extending the key generation and decryption algorithm definitions. Loosely speaking, generation of a decryption key from a master key is bound to a certain functionality $F$, such that $Dec(k_{d_F}, c) = F(x)$ with $Dec$ being the decryption algorithm, $k_{d_F}$ the decryption key for functionality $F$, $c$ the ciphertext and $x$ the corresponding plaintext. A typical scenario would be that the data owner encrypts all data and outsources it to a public HBC server. The data owner can then, using the master key, generate specific decryption keys $k_{d_F}$ for a certain functionality $F$ out of a family of functionalities $\mathcal{F}$. This $k_{d_F}$ can be given to clients, which can then learn the result of applying $F$ upon the encrypted data, but nothing else.

Such FE schemes include Identity-Based Encryption (IBE) [Sha85; BF03; Coc01], that replace the traditionally randomly generated public key with a key that is the function of the users identity. As such attributes that uniquely describe a user, such as a name and address, or his email address can be used as the public key. It must not be possible for the user to later on deny his connection to the same public key. In such a system public keys are by design linked to users and thus authenticated. There is no need for a Public Key Infrastructure (PKI) that manages, certifies and distributes public keys on request. Variants on IBE are fuzzy-IBE [SW05], which supports fuzziness in the public key. An often given use case is the utilization of biometrical data — which is by default fuzzy — as key material. Another variation is hierarchical-IBE [HL02; Yao+04; GS02], which allows to define hierarchical access control to data encrypted under the respective encryption system. Furthermore, having a PEKS scheme implies IBE schemes, as shown by Dan Boneh et al. [Dan+04].

Further schemes are related to and also covered by FE. These include Attribute-Based Encryption (ABE) [Goy+06], which allows fine-granular access control over encrypted data by defining access structures. Collusion attacks are thwarted and

users are able to derive new keys, which are more restricted in access than their own keys. Delegation is a property, which is found in some FE schemes, *i.e.* generating further restricted keys given the own decryption key. This is typically associated with Hierarchical Identity-Based Encryption (HIBE) [HL02; GS02] and ABE [Goy+06] or other predicate encryption systems that support delegation like presented by Shi and Waters [SW08].

Furthermore, access policies can be associated with the private decryption keys (key-policy ABE) or the ciphertext (ciphertext-policy ABE). Broadcast-encryption [FN94; AI09] extends the functionality of ABE schemes by introducing direct revocation, by which revocation of private keys can be performed without updating all other private keys. Searchable Encryption (SE) [Cur+06; KPR12], as described in Section 2.2 is also covered by FE, similarly to different predicate encryption schemes like [BW07; KSW08; SW08].

**Access Patterns** While guaranteeing high security under these assumptions, they still leak access and very likely search patterns [Liu+14]. Similar to the leakage of search patterns in the private key setting described in Section 2.2.1 and attacks building upon this information, leaked access patterns can be exploited to identify queries. Islam, Kuzu, and Kantarcioglu [IKK12] describe an attack based on revealed access patterns. Boneh et al. [Bon+07] present a system that does not leak access patterns, but is also rather inefficient as pattern hiding is bought by an square root sized overhead in the size of the document collection. Leakage of access patterns can even result in key recovery attacks.

The proven way to even hide access and therefore also search patterns is to use an ORAM [Gol87], which however is highly inefficient due to polylogarithmic overhead in all scheme parameters — that is communication, computation, storage and number of protocol rounds. Even the current, more efficient, lighter constructions are still unpractical [Cur+06]. Alternatives to hide access patterns for retrieval of documents are single-database PIR schemes first described by [KO97]. Ostrovsky and Rabani [OR07] show how to use such computational PIR schemes to search on data, even if considering continuously updated or streaming data. However, to hide access patterns, such schemes must touch the whole database for a single query, as the server would otherwise learn that the user is not interested in the untouched entries. Taking many queries of possibly different users together in a batch can amortize the per-query costs. A single run over the whole database would again touch every entry, but answer for all considered queries are generated at once, which was shown by Ishai et al. [Ish+04] and generalized in [Ish+06]. Further, specific optimizations for certain architectures were presented like generic keyword searches in computational single-server PIR schemes, optimized by Blass et al. [Bla+12b] towards efficient execution in the Cloud computing paradigm with the help of the widely used Map-Reduce model.

### 2.2.3. Authenticated Data Structures

When dealing with not-fully trusted parties that store and operate on data — as it is the case with searchable encryption, it might be useful to perform different sanity checks from time to time. These can include Proof of Data Possession (PDP) or Proof of Retrieval (PoR) for stored data, as well as authenticated data structures or in general verifiable computing for checking on correct evaluation of computational tasks.

Research into PDP [Ate+07], providing efficient detection of unauthorized changes, and PoR [JK07], guaranteeing correct retrieval even in case of unauthorized changes, quickly emerged into new research fields. PoR schemes typically employ erasure coding to correct a small amount of incorrect data and thus accommodate for small scale tampering.

While PDP and PoR schemes allow validation of fixed and static data, it is highly desirable to check and validate correct operation on remote data. One way to achieve this goal in general is to use verifiable computing [GGP10]. The idea is that a function $F: 0,1^n \rightarrow 0,1^m$ should be evaluated remotely. This function is represented as Yao's garbled circuit [Yao86], while all input is encrypted using FHE [Gen09]. First, the input $x$ is enhanced by random labels, which — depending on the input bits — transform to random labels for the output bits. The client can then, after retrieving the output bits and labels, extract $F(x)$ and check if the labels are correct for the received output. The chance of the worker to guess correct labels must be very small, so that the client is ensured that the output is correct. FHE is employed to hide the output labels from the worker. If the worker would see $F(x)$ together with the correct labels, he could also present $F(x)$ with the associated output labels for any different input $x'$, without the client being able to detect the misbehavior. The downside of this approach lies within the associated overhead of evaluating a garbled circuit over a FHE system.

Other approaches use authenticated data structures [Tam03], which allow operations upon them to be checked and verified. The Merkle hash tree [Mer90] is an example for a static data structure, which allows answers to queries to be checked. The construction does however not easily allow interaction with the data. Naor and Nissim [NN00] solved this issue by proposing authenticated trees, which can handle updates upon the stored data. Further authenticated tree-based protocols were presented by Buldas et al. [Bul+00], Gassko, Gemmell, and MacKenzie [GGM00], and Kikuchi et al. [Kik+99]. Simpler and thus more efficient constructions succeeding the authenticated trees were skip-list based protocols [GTS01; Mic01], which need no rebalancing operation to be performed, as balancing happens over time. Ranked based authenticated skip lists were the followup constructions which were optimized for storage size improvements and specifically designed to store files remotely [Erw+09].

## 2.3. Inference Control

Defining inference control in a broader sense might be done intuitively by stating that a privacy-preserving scheme should only output as few information as possible and as much as necessary. However in this sense most privacy-preserving schemes implement inference control. A stricter definition would require such a scheme to offer mechanisms for input or output regulation. Schemes employing input regulation will learn something about the input and be able to detect and mitigate — being active. Whereas output regulation works might also take information about the input into account, but could also work independent of the input. A possible approach for output regulation would be to employ anonymization techniques known from privacy-preserving data publishing.

**Output Inference Control Schemes** Output inference control can be achieved in several ways. The regulator can, for example, perturb the results, just before returning them to the querier. Perturbation can again be done using many different techniques, with differential privacy [Dwo06] being an actively studied and recently used example. It gained popularity for giving impossibility results on absolute privacy guarantees and providing relative privacy guarantees for a single user participating in a database or not. Next to perturbation, data aggregation, generalization and suppression are well known methods from the Privacy-Preserving Data Publishing (PPDP) research community. The final information that the output carries can be limited down to a single bit. This could be a binary answer to an arbitrary question like: "Is the distance between two inputs below a certain threshold?" (or within a given range). The functionality of calculating the distance in a privacy-preserving way can be delivered by any privacy-preserving distance calculation scheme. The (output) inference control functionality would be the range testing and binary answer generation. Domingo-Ferrer [Dom08] gives an overview over such methods, but not in the context of secure computation, but data publishing. As such there is no relation to input leakage from other parties or application of these techniques upon encrypted data.

**Input Inference Control Schemes** Input inference control requires the input of the privacy-preserving algorithm to be checked in any way to decide if and possibly what level of inference control actions should be applied. Of course to be able to decide input inference control prerequisites, the server must either learn some information about the input of one or more participating parties, that is the privacy-preserving scheme leaks information towards the server with the purpose of applying inference control. This can be done for HBC parties by actively providing input to the inference control allgorithm, like checksums, message authentication

codes or commitments. For malicious supplicants, the input to the inference control protocol must be validated to be correct, complete and up to date. This can be done using ZKPs to ensure input to the inference control prerequisites is properly derived from the input to the overall privacy-preserving scheme. To the best of our knowledge, these schemes are not studied together with secure computation and thus there is a lack of related work.

## 2.4. Symmetric Encryption Decrypted Homomorphically

As homomorphic cryptographic schemes are typically based upon randomized asymmetric cryptography, they inherit their drawbacks like slow overall performance, relatively large memory consumption and ciphertext expansion due to randomization, leading to transmission and storage overheads. All of these drawbacks are not really desired and it would be nice to at least partially circumvent them.

To increase storage and communication efficiency while using homomorphic encryption systems, a combination of symmetric and asymmetric cryptography — similar to the construction of a hybrid cryptographic system as used by [Cal+07] and [Kal98] — was proposed. In the hybrid cryptographic system case, an asymmetric system is used to encrypt a symmetric key $k_s$, which is used for a symmetric cryptographic system to encrypt the actual data. Through such a construction, properties from the asymmetric system are combined with the efficiency of symmetric cryptography.

The idea of combining a symmetric cryptosystem for efficient client side encryption and transmission with an homomorphic cryptographic system for outsourced processing was mentioned in several papers. Data is encrypted symmetrically before sending it to a remote server for further processing. The cipher used for this will be referred to as the *transmission*-cipher. It should have minimal ciphertext expansion and a very simple decryption circuit. The server received data encrypted under the transmission-cipher and reverses the transmission encryption (typically symmetrical encryption) *within* the homomorphic encryption using the homomorphically encrypted symmetric key. It obtains the data encrypted under the homomorphic system, which we will refer to as the *compute* cipher. Or simply stated, there is a transition function, which takes the symmetrically encrypted data and the symmetric key — homomorphically encrypted and outputs the same data encrypted under the homomorphic cryptographic system. The encrypted data will never be decrypted and plainly accessible at the server side.

The reasons given to motivate such a hybrid system are twofold. Brakerski, Gentry, and Vaikuntanathan [BGV11] brings up the idea of using bootstrapping

within a leveled homomorphic cryptosystem upon an AES circuit to save storage space until data must be securely processed. Next to just promoting the use of a symmetric system in front of an homomorphic one, [GHS12b; DHS14; LCT14; Che+13] use the AES decryption circuit as a benchmark to compare the performance of homomorphic cryptosystems against each other. Therefore, they test and optimize their leveled/fully homomorphic cryptosystems to evaluate the AES decryption circuit as fast as possible.

The generic FHE construction was improved regarding its expansion factor and operation costs by introducing the concept of batching [BGH13; GHS12a; Che+13], which allows storing a vector of plaintexts within a single ciphertext and operate on all elements like in the Single Instruction Multiple Data (SIMD) setting.

There are two different ways of utilizing batching when operating on Advanced Encryption Standard (AES) circuits homomorphically. We follow the naming given by [Che+13].

- *Byte-Wise Bitslicing* refers to splitting the 128 bit state into groups of 8 bit, with a single ciphertext carrying one bit of each group and thus 16 ciphertexts are necessary to store and evaluate a single AES block homomorphically.
- *State-Wise Bitslicing*, also known as fully bit-sliced stores each bit of the AES state in a different ciphertext, leading consequently to 128 ciphertexts to represent the state of a AES decryption circuit.

One of the fastest homomorphic AES decryption evaluations is reported by Lepoint, Coron, and Tibouchi [LCT14], requiring (amortized by decrypting many blocks in parallel) roughly 23 seconds per block. The overall runtime was about three and a half hours, using a public key of size 11 GB.

The compression for homomorphic ciphers (presented in Chapter 6) decodes a single bit in about *one millisecond* (using [NS98] with a 32 bit sized plaintext group order), or roughly 0.1 seconds for a symmetrically encrypted block of 128 bit. It must be kept in mind that these performance values from the literature on AES decryption circuit evaluation use a LHE system with a security parameter between 72 and 80 bit. Our system uses a much higher security parameter between 80 and 128 bit. Even more, our proposal scales much better with the security parameter, compared to the performance penalty received for increasing the security parameter for LHE or FHE. See also Chapter 6 for the detailed construction, security proof and discussion.

A few contributions try to improve the overall performance of the hybrid symmetric/homomorphic cryptographic system construction. To achieve a performance gain, they especially use ultra-lightweight symmetric cryptographic systems like PRINCE [Bor+12] to achieve better homomorphic decryption performance [SES14], SIMON [Bea+13] is used in [LN14] or Trivium [Can06] in [Hu13].

Increased performance then follows from the use of more efficient homomorphic systems, as well as from the simpler decryption circuits of the lightweight cryptographic systems. Canteaut et al. [Can+15] uses the block cipher family LowMC [Alb+15], Trivium and Kreyvium Canteaut et al. [Can+15], to construct a similar system, which they call homomorphic ciphertext compression.

Hu [Hu13] describe a scheme conversion framework in which an additive or somewhat homomorphic encryption scheme is combined with a multiplicative homomorphic scheme to evaluate more complex functions. More detailed, an additive homomorphic encryption scheme can be converted into a multiplicative encryption scheme, while a somewhat homomorphic encryption scheme can be converted to a multiplicative scheme and also back — through evaluation of the decryption function of the multiplicative scheme — to further extend the range of possible functions that can be evaluated homomorphically. Furthermore, a client side compression is described using a symmetric cipher at the client and homomorphic evaluation of the symmetric decryption circuit at the server. The used symmetric system is the light weight stream cipher Trivium [Can06].

Reduction of communication and storage costs is also investigated in [CK13], which combines public-key encryption schemes with somewhat or fully homomorphic encryption schemes. Data is first encrypted under the traditional public-key encryption scheme without any padding and later – on the server side – transformed into an encryption under the final somewhat homomorphic encryption scheme. As some asymmetric cryptographic system provide rather simple decryption circuits, transition from the transmission-cipher to the compute-cipher can also be efficient for pure asymmetric transmission-ciphers. If the first public-key scheme already provides homomorphic properties, the function family of the combined scheme can be enhanced similar to the results of [Hu13].

More loosely related, several other publications are also trying to increase the efficiency of very basic secure computation primitives. For example by trading off functionality for evaluation efficiency [ZW14], or combining garbled circuits with homomorphic encryption schemes [KSS09] to use the most efficient scheme for the next operations and perform costly transitions between them.

The schemes, which perform homomorphic decryption of the symmetric cipher, are instantiated to just be able to evaluate the decryption circuit, but nearly no further functionality. This increases the decryption performance, but leaves no space for further actual computations.

To understand the background of these decisions, one must look at the multiplicative depth of a circuit. The multiplicative depth of a circuit is defined as the length of the longest multiplicative path — only counting multiplications along the path — from any input to any output of the algorithm. The cryptographic

system must be configured and build to be able to handle this maximum multiplicative depth, while its efficiency typically decreases when the possible multiplicative depth is increased.

### 2.4.1. Fully Homomorphic Encryption Friendly Algorithm Design

FHE systems can be seen as a virtualization layer between the host CPU and a virtual CPU, providing oblivious instruction evaluation. This concept of a privacy-preserving virtual machine has been studied several times. This even goes into the design of a fully homomorphic cryptographic processor, offering instructions to perform fully homomorphic operations natively on the processor [BB13]. However specializing algorithms towards efficient execution on such virtual processors, or upon specific FHE or LHE systems is a research field still open for exploration and exploitation. This is mentioned as side notes at times [Fau+13], but not yet studied in depth.

Such a specialization follows the research of efficient algorithms for other restricted devices, like Application-Specific Integrated Circuits (ASICs), microcontroller or Field-Programmable Gate Arrays (FPGAs), for which lightweight and ultra-lightweight cryptosystems are designed and specifically challenged in projects like eSTREAM [RB08]. The same can apply to the construction of cryptosystems, which are optimized for efficient evaluation within FHE systems.

The argumentation for having optimized algorithms for virtual architectures follows the same line of arguments as for other real architectures. Single operations and construction techniques have a very different performance impact on different architectures. An example are conditional jumps and variable length loops, these are more or less efficiently implementable on architectures, which work on plain data, but cannot be used on architectures working on encrypted data. Program flow cannot be changed if the information for deciding is not available. However, encrypted architectures can simulate such conditional jumps by evaluating all possible paths. Such behavior of course quickly leads to very inefficient algorithms.

Another example are lookup-tables, which are used quite often in normal programs. However, due to the above mentioned issue of performing conditional jumps, lookups within tables are not easily implementable on encrypted architectures. To work around this, efficient logic is needed to replace the tables and deliver correct results. This is only feasible for small tables and is used to efficiently perform homomorphic evaluation of the AES decryption circuit by Doroz, Hu, and Sunar [DHS14]. They use a technique presented by Boyar and Peralta [BP10] from the area of logic minimization to get a low gate-count circuit for the AES decryption logic.

FHE, Somewhat Homomorphic Encryption (SHE) and LHE typically includes noise within its ciphertexts, which is low after encryption and grows together with the homomorphic operations performed. The amount by which the noise grows differs between homomorphic addition and homomorphic multiplication, with homomorphic multiplication resulting in a much larger growth rate than homomorphic addition. The noise level determines if the ciphertext can still be decoded correctly and thus sets limits on the maximum additive and multiplicative circuit depth, that a certain SHE or LHE can evaluate. As multiplicative multiplication increases the noise much faster, algorithms should mainly be optimized in their multiplicative circuit depth.

Lightweight algorithms intuitively seem to fit to the restrictions enforced by homomorphic cryptography, are however optimized for different goals. Looking at symmetric cryptographic schemes, one might be a low gate count as for SIMON and SPECK [Bea+13], or LED [Guo+11]. Another would be to achieve a high throughput to gate-size ratio like LBlock [WZ11], or low latency as offered by PRINCE [Bor+12]. The downside of having simple round functions, leading to small silicon footprints is the typically increased number of rounds performed for lightweight ciphers. An example is KATAN and KTANTAN proposed by De Cannière, Dunkelman, and Knežević [DDK09], which uses 254 rounds to achieve the desired security level of 80 Bits.

First optimizations in this regard are studied by Tillich and Smart [TS14], which look at the performance of different functions, construct their circuits and compare them regarding their evaluation within Multi-Party Computation (MPC) and FHE systems. Furthermore Albrecht et al. [Alb+15] construct the LowMC cipher family, which offers a reduced multiplicative size of the cryptographic primitive, making it favorable for MPC and FHE evaluation. Upon these constructions Canteaut et al. [Can+15] describes a system that uses the LowMC family to perform homomorphic ciphertext compression with increased efficiency.

# 3. Background

We are going to describe mathematical, cryptographical and otherwise technically necessary primitives used over the next chapters for constructing a privacy-preserving comparison system.

## 3.1. Edit Distance

A typical string comparison algorithm is the Levenshtein distance [Lev66], which is often also referred to as edit distance and describes the minimum number of insertions, deletions and substitutions needed to transform one string $s_1$ into another $s_2$. The result is a distance measure $d_{\mathrm{E}}$, which can easily be converted into a similarity score $s_{\mathrm{E}}$ between zero and one by: $s_{\mathrm{E}} = 1 - \frac{d_{\mathrm{E}}}{d_{\mathrm{E}_{\max}}}$.

$d_{\mathrm{E}_{\max}}$, *i.e.* the maximum distance between two strings, equals the length of the longer string and can thus be replaced by $d_{\mathrm{E}_{\max}} = \max(|s_1|, |s_2|)$ regarding the Levenshtein distance.

$$s_{\mathrm{E}} = 1 - \frac{d_{\mathrm{E}}}{\max(|s_1|, |s_2|)}$$

The algorithm for computing the Levenshtein distance belongs to dynamic programming and outputs more than just the actual distance between two strings. The original algorithm also generates a table with distances between all prefixes of $s_1$ and $s_2$. Table 3.1 depicts such a table for the strings *security* and *secrecy*. The green field on the bottom right gives the actual edit distance, while the red fields describe one possible edit path — which fixes the sequence of operations to transform one string into the other. As the edit path is constructed using a backtracking algorithm starting with the edit distance result, it depends on the actual backtracking algorithm which out of the possible edit paths is build.

| | | s | e | c | r | e | c | y |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| s | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| e | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| c | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| u | 4 | 3 | 2 | 1 | 1 | 2 | 3 | 4 |
| r | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 |
| i | 6 | 5 | 4 | 3 | 2 | 2 | 3 | 4 |
| t | 7 | 6 | 5 | 4 | 3 | 3 | 3 | 4 |
| y | 8 | 7 | 6 | 5 | 4 | 4 | 4 | 3 |

Table 3.1.: Table generated by original Levenshtein algorithm [Lev66] given equal weights for insertion, deletion and substitution.

**Edit Distance Extensions** Extensions of the Levenshtein distance introduce weighting of single operation or the swapping of consecutive characters, which can again also be weighted. Several optimizations were presented to calculate only necessary parts of the overall table to reduce the quadratic time complexity from $\mathcal{O}(n \cdot m)$ with $n = |s_1|, m = |s_2|$ being the lengths of the strings. Assuming a maxium edit distance for two strings $d_{\text{Emax}}$, Ukkonen [Ukk85] defines a method to only compute values around the table diagonal. This algorithm has complexity $\mathcal{O}(d_{\text{Emax}} \cdot \min(n, m))$.

Finally a Levenshtein automata [SM02] is a Finite State Machine (FSM), which is build given an input string $s_1$ and a maxium edit distance $d_{\text{Emax}}$ and accepts any other string $s_2$ with an edit distance up to $d_{\text{Emax}}$.

## 3.2. Q-Grams

A string can be converted into and represented by a set of shorter strings. This allows set operations to estimate results of string operations. Several ways exist to generate this representative set, depending on the properties that should be preserved from the original string. We use the notion of $q$-grams, which defines items over the input string and selects $q$ consecutive items as a $q$-gram. Different domains use deviating definition for what actually an *item* is. As an example, *gram* can refer to a character, word, syllable or even base pair or nucleic acid in biology. We will always use gram to denote a single character of a string. As such a $q$-gram is a substring of $q$ consecutive characters.

Let $s_{[i,j]}$ be the substring of a string $s$ starting at position $i$ and ending at (including) position $j$, thus $s_{[i,(i+q-1)]}$ — or just short $s_i$ — defines the $q$-gram

starting at position $i$ in $s$. For a string of length $n = |s|$, $n_q = n - q + 1$ $q$-grams exist. The complete set of all possible $q$-grams is then defined as $S = \bigcup_{i=1}^{n_q} s_i$.

The resulting elements in $S$ are independent of their original position, information about repetitions within strings and permutations between substrings is not captured anymore by the set representation. To accommodate for this, the $q$-grams are prefixed with their original position within the string $s$. A *positional* $q$-gram is then defined as $(i, s_i)$.

Using this construction, characters at the beginning and end of the string are in less $q$-grams than characters in the middle of the string. An example: The first and last character is only present in a single $q$-gram — the first and last $q$-gram. On the other hand, for strings with length $n \geq 2q - 1$, characters in the middle of $s$ are present in $q$ $q$-grams.

To even out the underrepresented characters at the beginning and end of the sequence, an extension of $q - 1$ identical characters, which are not part of the original alphabet, is appended on both sides. Positional $q$-grams on extended strings were introduced by Gravano et al. [Gra+01].

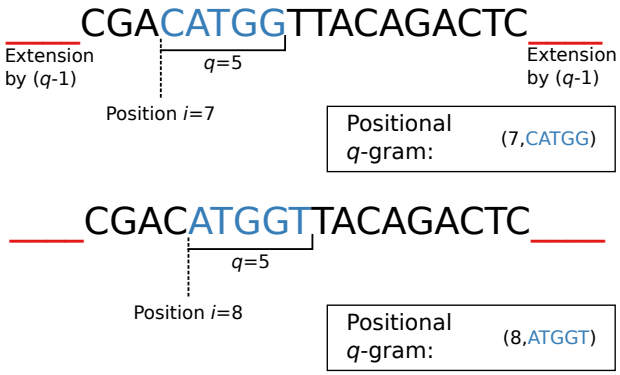



Figure 3.1.: A sliding window moving over a string to generate $q$-grams. The string is prefixed with $q-1$ non-alphabet symbols and the position is attached to create *positional* $q$-grams.

Figure 3.1 visualizes the sliding window that moves over the extended input string to read out positional q-grams.

The concept of $q$-grams is related to $k$-mers and $n$-tuples. A generalization of $q$-grams are skip-grams. As such a $k$-skip-$q$-gram also consists of $q$ items, which must however not be consecutive and may be up to $k$ positions away from each other.

Selecting the correct value for $q$ is a non-trivial task. From a "feeling" point of view, it seems intuitive that the longer the string $s$ is, the bigger $q$ should be. As

such, for representing single words, $q = 2$ could be fitting, while sentences might achieve better results with something like $q = 4$ and documents like scientific publications or patents could be best represented around $q = 9$.

We will however eliminate the need to choose $q$ on our own, as we will be using a variable length gram algorithm, which is trained using a reference dataset.

## 3.3. VGRAM

Different techniques for selecting a feasible value for $q$ were proposed, however instead of using a single fixed value for $q$, we use an algorithm that choses the length of the current gram dynamically, based on its frequency within a previously given reference dataset. The algorithm is called VGRAM and was proposed by Li, Wang, and Yang [LWY07].

Following the VGRAM algorithm, a range of possible values for $q$ is defined by $[q_{\min}, q_{\max}] \subset \mathbb{Z}$. All possible positional grams within this range are generated and its frequency over a predefined dictionary is recorded. A frequency threshold then defines the desired frequency a gram should have, which is leveraged during the actual gram generation. The gram length is chosen on-the-fly while constructing grams in such a way that the frequency of the generated gram is as close as possible to the frequency threshold.

The final grams set does not include positional information to allow grams with insertions or deletions on previous positions and thus changed indices to be matched. Li, Wang, and Yang [LWY07] also defines a relation between the edit distance and the set intersection cardinality upon a VGRAM set. This is done by giving a lower bound on the number of common grams between two strings converted via VGRAM using the same dictionary. Further a upper bound on the Hamming distance between bit vectors, representing the VGRAM sets is given.

We used the "Human Mitochondrial Genome Database" [IG06] as a reference dataset to train the algorithm. The resulting grams — chosen by the VGRAM algorithm — are similarly common within the representative training data and thus should yield comparable frequencies for the actual data. This means larger $q$ values are used for more frequent grams, whereas shorter grams are preferred if a sub-string is not well represented within the dictionary.

## 3.4. Bloom Filter

A Bloom filter is a probabilistic data structure representing a set. Elements can be added to the data structure and member tests can be performed. Checking for an element is probabilistic due to the design of the filter.

Let $b$ be an array of bits of length $n$ and $b_i$ the $i$-th value within the array with $i \in [1, n]$. Further let $h_1 \dots h_k$ be $k$ hash functions, with uniformly distributed output in $[1, n]$. For initialization set $b_i = 0 : \ \forall i \in [1, n]$.

To add an element $e$ to the filter, evaluate all $k$ hash functions on $e$ and treat the results as indices for $b$ to set these positions to one. Set $b_{h_j(e)} = 1 : \ \forall j \in [1, k]$.

A member test for element $e'$ is performed by evaluating all $k$ hash functions upon $e$ and checking the referenced positions in $b$. If at least one of the positions $b_{h_j(e')}$ is set to zero, the element has not been added to the Bloom filter before. If all bits are set to one, however, one cannot be sure if a previous insertion of $e'$ set these, or if these are hash collisions with with other elements.

Using these operations a set is represented by adding all set elements to the filter. Depending on the filter parameters, Christensen, Roginsky, and Jimeno [CRJ10] define the probability $p_{\text{fp}}$ that a false-positive member test occurs — *i.e.* that an element is falsely identified as being added to the filter before — by equation (3.1). $l$ carries the number of inserted elements, $k$ represents the number of hash functions used and $n$ specifies the size of the binary array.

$$p_{\text{fp}} = \left(1 - \left(1 - \frac{1}{n}\right)^{kl}\right)^k \tag{3.1}$$

$\left(1 - \frac{1}{n}\right)^{kl}$ specifies the probability that a single bit is still zero after $l$ elements have been added to the filter of length $n$ using $k$ hash functions. This formula can be transposed to calculate the required length $n$ of a Bloom filter given the desired false-positive probability $p_{\text{fp}}$ for the last inserted element and the actual number of elements to be inserted $l$. The equation (3.2) then specifies the required bit length to achieve a certain false-positive probability.

$$n = \frac{-1}{\sqrt[kl]{\left(1 - \sqrt[k]{p_{\text{fp}}}\right)} - 1} \tag{3.2}$$

As specified above, Bloom filters can be used for membership tests and estimations of set cardinality, Papapetrou, Siberski, and Nejdl [PSN10] analyzed how the number of hash functions used for constructing the Bloom filter influences the accuracy of estimated results. It was concluded, that the optimal number of hash functions to do cardinality estimation is one. Based on this — as we also want to do cardinality estimation — we fix $k = 1$ and only use a single hash function to build and query Bloom filters throughout the rest of the thesis.

By having $k = 1$ Equation (3.1) simplifies to $p_{\text{fp}} = 1 - (1 - 1/n)^l$ and Equation (3.2) to $n = \left(\sqrt[l]{1 - p_{\text{fp}}} - 1\right)^{-1}$ respectively.

## Bloom Filter Operations

Due to the structure of a Bloom filter, it can be easily used to approximate set operations for represented sets. Given two Bloom filters $b_1, b_2$ representing the sets $S_1, S_2$, the primitive set operations *union* $S_1 \cup S_2$, *intersection* $S_1 \cap S_2$ and *symmetric difference* $S_1 \triangle S_2$ are estimated using the binary *or* $b_1 \vee b_2$, *and* $b_1 \wedge b_2$ and *xor* $b_1 \oplus b_2$ operations between the binary representations of the Bloom filters $b_1$ and $b_2$.

The *absolute set complement* denoted $S_1^C$ requires a universe $U$ to be defined, with $S_1 \subseteq U$. A possible equivalent for the Bloom filter $b_1$ can be found in the binary negation $\neg b_1$, but it must be used carefully. If there is a universe $U$, it is very unlikely that $\neg b_1$ is a good approximation of $S_1^C$. However, the absolute set complement can be used to derive a good approximation for the *relative complement* $S_1 \setminus S_2 = S_1^C \cap S_2$ through $(\neg b_1) \wedge b_2$ or short $b_2 - b_1$.

Given the bit-wise *or* operation $\vee$ upon Bloom filers, one can see that the Bloom filter $b_\vee = b_1 \vee b_2$ is identical to the Bloom filter $b_\cup$ derived from $S_1 \cup S_2$. This is however not true for set intersection and the absolute complement, as described above. Intersection $\wedge$, relative complement $-$ and symmetric difference $\oplus$ may produce different representations, compared to applying the set operation upon the original sets and deriving the Bloom filter afterwards.

A simple example for $k = 1$ is $S_1 = \{a\}$ and $S_2 = \{b, c\}$ with $b$ producing a collision with the hashed value of $a$. Surely $S_1 \cap S_2 = \emptyset$ and the cardinality of the thereof generated Bloom filter is zero $|b_\cap| = 0$, but due to the collisions generated, $b_\wedge = b_1 \wedge b_2 = b_1$ the intersection equivalent produces a non-empty Bloom filter. Further more, a membership test on $b_\wedge$ regarding the element $a$ or $b$ returns true, which is clearly wrong.

## 3.5. Homomorphic Encryption

An homomorphic encryption scheme, is a cryptographic encryption schemes, which exhibit at least one homomorphism between the plaintext group and ciphertext group.

A client wants to outsource computation to a server and uses a homomorphic cryptographic system to hide the sensitive data from the less trusted compute server. He chooses a cryptographic system that can evaluate the desired functionality homomorphically and encrypts all private input data using this scheme. All necessary plaintext data together with the generated encrypted ciphertexts are send to the compute server. The server performs all necessary homomorphic operations and sends the result back to the client.

**Definition 3.1: Homomorphic Encryption Scheme**

A homomorphic encryption scheme (HE) is a quadruple $\mathcal{H} = (\mathsf{KeyGen}_\mathcal{H}, \mathsf{Enc}_\mathcal{H}, \mathsf{Dec}_\mathcal{H}, \mathsf{Eval}_\mathcal{H})$ of polynomial time algorithms. $\mathsf{KeyGen}_\mathcal{H}(1^\kappa) = (\mathsf{sk}, \mathsf{pk})$ is the key generation function, which takes a security parameter $\kappa$ and outputs the secret key $\mathsf{sk} \in K_S$ and public key $\mathsf{pk} \in K_P$. $\mathsf{Enc}_\mathcal{H}\colon K_P \times P_\mathcal{H} \to C_\mathcal{H}$ denotes the encryption function with $\mathsf{Enc}_\mathcal{H}(\mathsf{pk}, m) = c$ and equivalently $\mathsf{Dec}_\mathcal{H}\colon K_S \times C_\mathcal{H} \to P_\mathcal{H}$ the decryption function with $\mathsf{Dec}_\mathcal{H}(\mathsf{sk}, c) = m'$. Plaintexts are $m, m' \in P_\mathcal{H}$ and $c \in C_\mathcal{H}$ is a ciphertext. $\mathsf{Eval}_\mathcal{H}\colon K_P \times \mathcal{F} \times C^*_\mathcal{H} \times P^*_\mathcal{H} \to C_\mathcal{H}$ performs an homomorphic evaluation of some functionality. Let $\mathsf{Eval}_\mathcal{H}(\mathsf{pk}, F, \bar{c}, \bar{p})$ be such an evaluation for a function $F \in \mathcal{F}$ given some ciphertexts $\bar{c} \in C^*_\mathcal{H}$, and some plaintexts $\bar{p} \in P^*_\mathcal{H}$. Decryption and evaluation must be correct, that is $\mathsf{Dec}_\mathcal{H}(\mathsf{sk}, \mathsf{Enc}_\mathcal{H}(\mathsf{pk}, m)) = m$ and $\mathsf{Dec}_\mathcal{H}(\mathsf{sk}, \mathsf{Eval}_\mathcal{H}(\mathsf{pk}, F, \bar{c}, \bar{p})) = F(\bar{m}, \bar{p})$.

The function family $\mathcal{F}$ depends on the choice of the actual encryption scheme and its supported homomorphisms. In case $\mathsf{Eval}_\mathcal{H}$ needs no plaintext input, we just use $\mathsf{Eval}_\mathcal{H}(\mathsf{pk}, F, \bar{c})$ and imply an empty plaintext vector. For example, Partly Homomorphic Encryption (PHE) schemes can be split into Additive Homomorphic Encryption (AHE) [Pai99; NS98] and Multiplicative Homomorphic Encryption (MHE) schemes [ElG84; RSA78]. Schemes that support both homomorphisms, addition and multiplication can be split into Somewhat Homomorphic Encryption (SHE), Leveled Homomorphic Encryption (LHE) [LCT14] and Fully Homomorphic Encryption (FHE) [Gen09; Bra12]. They typically differ in the operational depth of the function that can be evaluated, especially depending on the maximum path length counting consecutive multiplications throughout the evaluation of the function $F$.

*Asymmetric* homomorphic cryptographic systems need to be randomized and thus indeterministic to achieve the notion of semantic security [GM84]. Semantic security implies that an adversary, given a ciphertext, gains no information about the corresponding plaintext. Randomization is achieved using a random value as input to the encryption function $\mathsf{Enc}_{\mathcal{H}}$, however for the sake of brevity this randomization parameter is omitted. In case this parameter is relevant for the scheme, it will be explicitly included — typically as $r, r', r'', r'''$ — which are independent uniform random variables, unless specified differently. To increase readability and ease of presentation further the public key parameter can also be omitted if the context is clear and the shorter notation $E(m) = \mathsf{Enc}_{\mathcal{H}}(\mathsf{pk}, m)$ is used. This notation might also carry the randomization parameter, *i.e.* $E(m, r)$.

Next to PHE exist schemes that offer increased functionality. These are denoted SHE and LHE and offer the possibility to evaluate both operations homomorphically (addition and multiplication) while being restricted to low multiplicative-depth circuits. The difference between them is that LHE guarantees a certain multiplicative depth that can be evaluated properly, whereas SHE does not provide such guarantees. If $F$ uses too many consecutive multiplications for a specific instantiation of an SHE or LHE system, the correctness of decryption is not guaranteed anymore. In contrast to these schemes, the function family $\mathcal{F}$ for a FHE contains all boolean functions of arbitrary (multiplicative) depth.

## 3.6. Encrypted Bloom Filter

A Bloom filter can be used to perform cardinality estimation or run membership tests over a set. It further allows to perform equivalences of set operations on two Bloom filter representations. These basic set operations like union ($\cup$), intersection ($\cap$) or difference ($\setminus$) can be used to construct metrics over sets like the Jaccard index $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$ or Sørensen $S(X, Y) = \frac{2|X \cap Y|}{|X|+|Y|}$. Such operations upon Bloom filters were discussed in Section 3.4.

Representations of strings or character sequences were constructed in Sections 3.3 and 3.4 using the VGRAM [LWY07] algorithm and Bloom filters. However despite Bloom filters being a very efficient data structure for performing the desired operations, they are themselves not a good way to preserve the confidentiality of the original strings. Several publications including [KS15; Kuz+11; Nie+14] show that a lot of original information can be reconstructed given a typical Bloom filter. Even so they also propose Bloom filter configurations which limit the leakage according to their attack scenario, the privacy guarantees of a Bloom filter [BBL12] are too weak for our concern.

Thus we construct encrypted Bloom filters to preserve privacy of the filter content. An additively homomorphic cryptosystem is used as described in Section 3.5.

A homomorphic cryptosystem offers at least one homomorphic property to evaluate an operation $\oplus$ on the ciphertext, which translates into applying the equivalent operation $+$ on the plaintext. We will use an Additive Homomorphic Encryption (AHE) — *i.e.* the one introduced by Naccache and Stern [NS98] — which is probabilistic and offers semantic security.

Let $E(x, r)$ denote the encryption of a value $x$ using a fresh random value $r$ for each encryption, this additively homomorphic system has the following properties:

$$E(x, r) \cdot E(y, s) = E(x + y, rs)$$
$$E(x, r)^y = E(xy, r^y)$$

Furthermore, let $E(x, r)^{-1}$ denote the calculation of the multiplicative inverse upon $E(x, r)$, found through executing the extended euclidean algorithm, which is by the homomorphism definition the encryption of the additively inverse plaintext. This results in $E(x, r)^{-1} = E(-x, r)$ and can be used to calculate a difference between two encrypted values.

To multiply an encrypted plaintext with a negative factor $-z$, first the multiplicative inverse of the encrypted value is calculated and then multiplied using the positive factor.

$$\begin{aligned} E(x, r)^{-z} &= E(x, r)^{-1 \cdot z} \\ &= E(-x, r)^z \\ &= E(-xz, r^z) \end{aligned}$$

An encryption of a Bloom filter $b$ with length $n$ is constructed by encrypting every bit in $b$ separately, storing the resulting $n$ values in a new array $c$ with equal length.

Figure 3.2 gives an example for a possible $q$-gram to Bloom filter mapping, followed by an encryption using a homomorphic cryptosystem. Similarly equation (3.3) defines the encryption of a Bloom filter.

$$c_i = E(b_i, r) : \ \forall i \in [1, l] \text{ and fresh } r \tag{3.3}$$

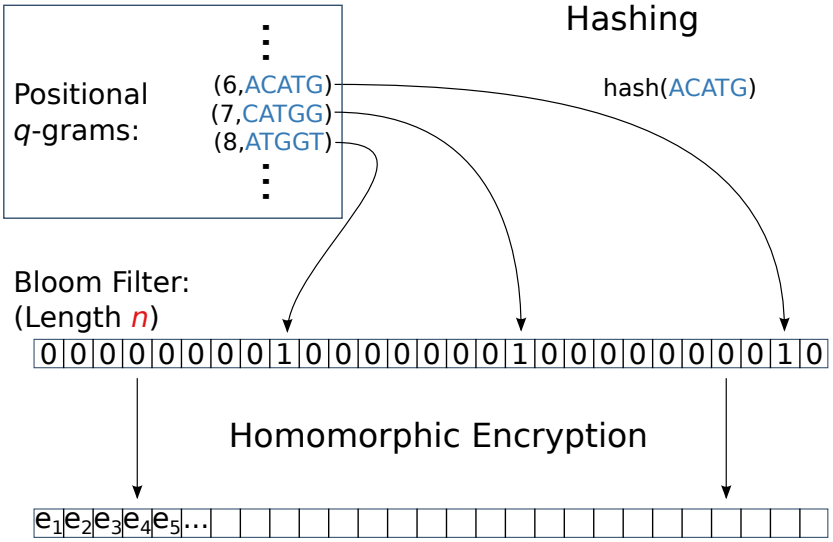



Figure 3.2.: Construction of a Bloom filter over $q$-grams and conversion to an encrypted Bloom filter.

## 3.7. Error-Correcting Codes (ECCs)

ECCs are commonly used to correct errors occurring during transmission. Let $x$ be a bit string of length $l'$ bits. We call $x$ the information word. An ECC will then transform the information word into a codeword of length $l > l'$ bits containing $(l - l')$ check bits. The resulting codeword $y$ is typically transmitted by the sender over an error-prone communication channel. The transmission might be faulty and the recipient receives $y'$. On the receiving end the participant can recover $x$ from $y'$ as long as the distance between $y$ and $y'$ is below a certain threshold $t$. The performance of a code and, therefore, the threshold $t$ is described by the minimum Hamming distance $d_{\mathrm{H}_{\min}}$, defined as the minimum distance among all possible distinct pairs of $2^{l'}$ codewords in the code alphabet. The maximum number of correctable errors can be calculated with $t = \lfloor (d_{\mathrm{H}_{\min}} - 1)/2 \rfloor$. Of course, a high performance $t$ requires a high amount of check bits $(l - l')$.

In chapter 5 the investigations are based on $\left(l, l', d_{\mathrm{H}_{\min}}\right) = \left(2^m, \sum_{i=0}^{1} \binom{m}{i}, 2^{m-1}\right)$ first-order Reed-Muller codes as defined by Muller [Mul54] and Reed [Ree54]. Furthermore, we use a first-order code, such that we only have first-order decoding equations, *i.e.*, each decoding equation sums two and only two received bits. Most importantly, we use *modified* Reed-Muller codes. We modify the code, such that it is shortened and has an odd number of decoding equations for each information word bit as far as this is possible. This has the purpose that each word $y'$ has

exactly one associated information word $x$. In particular, we avoid by this construction decoding failures where the decoding equations result in an equal number of 0s and 1s.

Let $\mathsf{Enc}_{\text{ecc}}\colon \{0,1\}^{l'} \to \{0,1\}^{l}$ denote the encoding function, which takes an information word $x$ and outputs the corresponding codeword $y = \mathsf{Enc}_{\text{ecc}}(x)$ for the given ECC. Likewise $\mathsf{Dec}_{\text{ecc}}\colon \{0,1\}^{l} \to \{0,1\}^{l'}$ denotes the decoding function, that decodes a received bit sequence and recovers the information word $x' = \mathsf{Dec}_{\text{ecc}}(y')$ (error-correction is performed implicitly during decoding). In general decoding may fail, but as we use a shortened Reed-Muller code with an odd number of decoding equations, we are always able to decode. The ECC is utilized in the proposed scheme in Chapter 5 in a way that the encoding functionality is only used to construct reference points, but the main focus lies upon decoding. However, the notion of a decoding error — that is decoding to an information word $x'$ different from the originating information word $x \neq x'$ — is not of importance for our work.

### Reed-Muller

As described above, we are using first-order Reed-Muller codes [Mul54; Ree54]. We therefore describe encoding and decoding, as well as our introduced shortening of the code in order to eliminate decoding failures.

A first-order Reed-Muller code $\mathcal{R}(1,m)$ with $m > 0$ is a $\left(2^m, m+1, 2^{m-1}\right)$ linear code. Following this, each information word has length $l' = m+1$ and each codeword has length $l = 2^m$. All codewords have a Hamming weight of $2^{m-1}$, except the codeword $\mathbf{0} = (0,\ldots,0)$ generated from the information word $(0,\ldots,0)$ and the codeword $\mathbf{1} = (1,\ldots,1)$ generated from the information word $(1,0,\ldots,0)$. Let

$$G_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

be the generator matrix of the first-order Reed-Muller code $\mathcal{R}(1,1)$. The generator matrix for the first-order code $\mathcal{R}(1,m+1)$ is then defined as

$$G_{m+1} = \begin{pmatrix} G_m & G_m \\ 0,\ldots,0 & 1,\ldots,1 \end{pmatrix}.$$

Encoding follows the normal encoding of linear codes using a vector-matrix multiplication. An element from the finite field $x \in \mathbb{F}_2^{m+1}$ is encoded to $y = \mathsf{Enc}_{\text{ecc}}(x) = xG_m$ with $y \in \mathbb{F}_2^{2^m}$. Reed [Ree54] defines a simple decoding algorithm using majority voting. This decoding algorithm guarantees correct decoding in case of up to $t$ bit errors. Its basis is again the generator matrix $G_m$, which has $l'$ rows, denoted by $r_i$ $(0 \leq i \leq m)$. Let $x'_i$ be a binary value at position $i$ of a

bit sequence $x'$, which is the result of the decoding algorithm applied to a possibly corrupted bit sequence $y'$ of length $l$ bits. From now on we will purely focus on the decoding algorithm of the ECC and are thus not interested in (and even do have a notion for) the original information word within our application. Due to this we use $x$ to describe the result of the decoding algorithm instead of the typically used $x'$. Similarly we use $y$ as the bit string that is input to the decoding function $\mathsf{Dec}_{\mathrm{ecc}}$.

In case of a first-order Reed-Muller code, for $m$ bits of the decoded information word $x$, specifically for $x_i, (1 \leq i \leq m)$, $2^{m-1}$ decoding relations are obtained from the associated generator row $r_i$. For example, the Reed-Muller code $\mathcal{R}(1,4)$ has the associated generator matrix:

$$G_4 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{matrix}.$$

Let $y_j$ denote the $j$-th bit $(0 \leq j < l)$ from the received bit sequence $y$, which should be decoded. If $y$ was not corrupted and therefore is a codeword, then all following terms for a certain $i$ evaluate to the same result. The $2^{m-1}$ relations for $x_i$ (following the row $r_i$ and $1 \leq i \leq m$) are then:

$$\begin{aligned}
x_1 &= y_0 \oplus y_1 = y_2 \oplus y_3 = y_4 \oplus y_5 = y_6 \oplus y_7 = y_8 \oplus y_9 = y_{10} \oplus y_{11} = y_{12} \oplus y_{13} = y_{14} \oplus y_{15} \\
x_2 &= y_0 \oplus y_2 = y_1 \oplus y_3 = y_4 \oplus y_6 = y_5 \oplus y_7 = y_8 \oplus y_{10} = y_9 \oplus y_{11} = y_{12} \oplus y_{14} = y_{13} \oplus y_{15} \\
x_3 &= y_0 \oplus y_4 = y_1 \oplus y_5 = y_2 \oplus y_6 = y_3 \oplus y_7 = y_8 \oplus y_{12} = y_9 \oplus y_{13} = y_{10} \oplus y_{14} = y_{11} \oplus y_{15} \\
x_4 &= y_0 \oplus y_8 = y_1 \oplus y_9 = y_2 \oplus y_{10} = y_3 \oplus y_{11} = y_4 \oplus y_{12} = y_5 \oplus y_{13} = y_6 \oplus y_{14} = y_7 \oplus y_{15}
\end{aligned}$$

Figure 3.3 shows the connection between the structure of the generator matrix rows and the decoding relations. The relations for $\mathcal{R}(1,4)$ $(0 \leq j \leq 7)$ can also be expressed as:

$$\begin{aligned}
x_1 &= y_{2j} \oplus y_{2j+1} \\
x_2 &= y_{4\lfloor j/2 \rfloor + (j \bmod 2)} \oplus y_{4\lfloor j/2 \rfloor + (j \bmod 2) + 2} \\
x_3 &= y_{8\lfloor j/4 \rfloor + (j \bmod 4)} \oplus y_{8\lfloor j/4 \rfloor + (j \bmod 4) + 4} \\
x_4 &= y_{(j \bmod 8)} \oplus y_{(j \bmod 8) + 8}
\end{aligned}$$

For a first-order Reed-Muller code $\mathcal{R}(1,m)$, each of the $m$ terms $T_{i,j}$ for every $x_i$ can thus be generalized to

$$T_{i,j} = y_{2^i \lfloor j/2^{i-1} \rfloor + (j \bmod 2^{i-1})} \oplus y_{2^i \lfloor j/2^{i-1} \rfloor + (j \bmod 2^{i-1}) + 2^{i-1}} \tag{3.4}$$

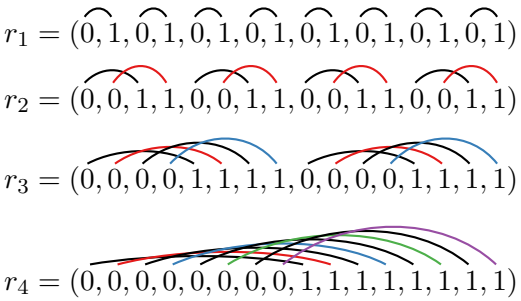



Figure 3.3.: Selection of Reed-Muller decoding relations following the generator matrix $G_4$. Example taken from Reed [Ree54]. Different colors and heights of arcs are only to ease following an arc and recognize related elements.

for $(1 \leq i \leq m, 0 \leq j < 2^{m-1})$. If $y$ is a codeword, then all terms $T_{i,j}$ for a fixed $i$ evaluate to the same result. Put differently, all relations given for $x_i$ hold. Note that for a certain $x_i$, no bit $y_j$ is used more than once within all related terms $T_{i,j}$. This ensures that in case of an bit error, only a single term $T_{i,j}$ evaluates to a wrong value, *i.e.* $m - 1$ values would still be correct. Bit-wise decoding is done by taking a majority vote of the term results $T_{i,j}$ for each $x_i$. A decoding failure happens in case the majority vote fails, that is if half of the terms evaluate to 0 and the other half to 1.

The described mechanism only works for $1 \leq i \leq m$, but not for $i = 0$. So decoding of the information word bit $x_0$ must be handled differently. In order to decode the remaining bit, the received bit sequence $y$ must be reduced by all decoded bits multiplied with their respective generator matrix row as follows:

$$y' = y \oplus \sum_{i=1}^{m} x_i r_i.$$

$y'$ describes the reduced bit sequence necessary for the last step of the decoding algorithm, not the possibly error-prone received bit sequence, as it was used during the description of general Error-Correcting Codes (ECCs). The sum operator $\sum$ is using the $\oplus$ operator for additions. The final bit $x_0$ is then again decoded using a majority vote over the bits in $y'$. Using the Hamming weight $w_\mathrm{H}$ of $y'$, decoding $x_0$ can specifically defined as:

$$x_0 = \begin{cases} 1 & \text{if } w_\mathrm{H}(y') > 2^{m-1} \\ 0 & \text{else} \end{cases}.$$

**Shortened Reed-Muller Decoding** The Reed-Muller decoding algorithm, as described above, can result in decoding failures in case $w_{\mathrm{H}}(\sum_{j=1}^{2^{m-1}} T_{i,j}) = 2^{m-2}$ for any fixed $i$. We however want to have no decoding failures, but still use the Reed-Muller code for its high error-correction capability and simple decoding function. In order to avoid all decoding failures, we simply use an odd number of terms $T_{i,j}$ for each $x_i$ by limiting $j$ to $0 \leq j < 2^{m-1} - 1$. In other words, we just ignore the last term for all decoded bits. This eliminates all decoding failures, as a majority vote over an odd number of votes will always succeed. The same technique is applied to the decoding of the final bit $x_i$, the last bit of the bit sequence $y'$ is not used for the final majority vote.

As a result the number of correctly decoded bits is as least as high as with the original decoding function, while the number of false decodings might increase. All previous decoding failures are decoded either correctly or incorrectly, depending on the correctness of the ignored term $T_{i,2^{m-1}}$ and the last bit of $y'$.

## 3.8. Commitments

In a commitment scheme a committer binds itself to a certain value without revealing this value. It consists for two phases, the commitment stage, in which a committer fixes a value $m$ he wants to commit to. The commitment must ensure the *binding* property — it must not be possible to change the committed value afterwards. Further confidentiality for the committed value must be ensured to fulfill the *hiding* property. The receiver of the commitment must not be able to learn anything about the committed value.

Generating a commitment can be done using a collision resistant hash functions $f(m)$. However, committing to a single bit or low entropy values is not be possible in such a way directly. Therefore Brassard, Chaum, and Crépeau [BCC88] proposes a scheme to commit to bits by also committing to a large enough random number $r$ as $f(m, r)$.

## 3.9. Zero Knowledge Proofs

A Zero Knowledge Proof (ZKP) is typically an interactive proof system between two parties $u_p, u_v$, which has certain distinctive properties. Specifically it must be *complete*, *sound* and *zero-knowledge*. It is constructed upon an commitment scheme as it was discussed in Section 3.8.

The *Completeness* property requires that for a honest prover $u_p$ and a honest verifier $u_v$, the overall proof must return a positive result with overwhelming probability. *Soundness* requires that for a malicious prover and a honest verifier the probability to get to a positive result must be negligible in the security parameter. Further, the zero-knowledge property requires that the verifier learns nothing except whether some statement that the prover wants to prove is true.

A typical round for an interactive ZKP is divided into three steps, as shown by Figure 3.4.

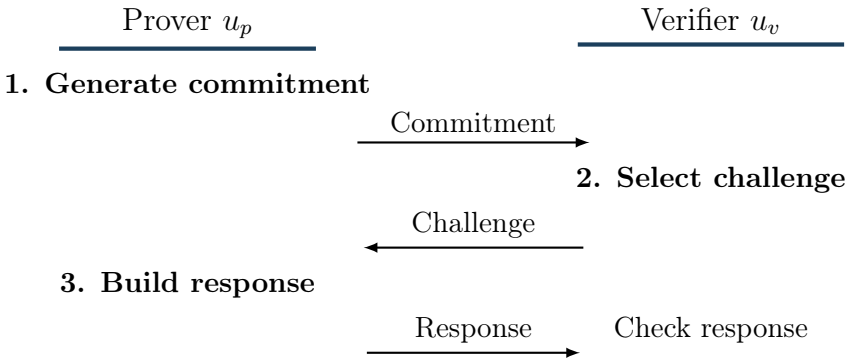
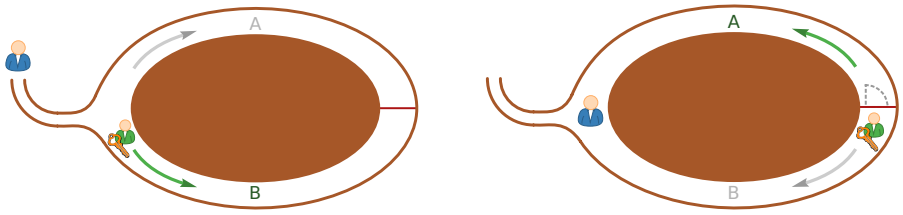



Figure 3.4.: Schematic overview of a generic zero-knowledge round, represented by the three typical steps.

Figure 3.5 shows a visualization for a protocol to perform a ZKP. Two users are standing in front of a cave, which has one entrance that goes around a corner and then splits into two different passages, which are connected only via a locked door. From the entrance to the cave one cannot see the parting of the way into the two passages. The task for user $u_p$ (the *green* user, or prover) is to prove to user $u_v$ (the *blue* user, or verifier) that he possesses a certain key to open the locked door that connects both passages.




(a) Commitment (b) Challenge

Figure 3.5.: A visualization of the "Strange Cave of Ali Baba" from Quisquater et al. [Qui+89]

Both stand outside the cave. In the commitment phase (Figure 3.5(a)) $u_p$ randomly chooses way $A$ or $B$ to follow till the end, while $u_v$ stays outside of the

cave. $u_v$ did not see which way $u_p$ was taking. Next, $u_v$ enters the cave till he arrives at the fork. He then — in the challenge stage, Figure 3.5(b) — selects a random passage that $u_p$ should return from and calls for him. If $u_p$ returns from the proper passage, this round succeeded. They will both leave the cave and repeat the experiment with fresh random decisions.

$u_p$ has a probability of 0.5 to cheat on each round, after $l$ consecutive successful rounds the chance of $u_p$ cheating is therefore reduced to $2^{-l}$. After several rounds, this probability is so low, that $u_v$ stops the ZKP and accepts the overall proof.

A proof of knowledge is an often used example for a ZKP. To a public value $x$ — known to the prover and verifier — belongs a specific secret $y$ via the relation $(x, y) \in R$. The prover then proves in zero-knowledge that he knows the corresponding $y$ to a certain $x$ without revealing $y$.

Even though the chances for the prover cheating and not getting caught can be made arbitrarily low, a ZKP never gives a probability of 1 that the proof was genuine and the prover didn't cheat.

Also *non-interactive* ZKPs systems exist, for which the proof can be validated offline, what makes these proves repeatable. Positive and negative consequences follow from that. On the positive side, it can be used to convince third parties, as they have all the information necessary for validating the proof themselves. On the other hand — as the proof can be copied — it can be represented as someone else's proof.

This constitutes in relay and Man-In-The-Middle (MITM) attacks, where the actual prover and verifier have at least one more intermediate party, which they cannot detect. Beth and Desmedt [BD91] gave some possibilities to solve this issue, while Cramer and Damgård [CD97] proposed the use of the OR-protocol [CS94], which mitigates MITM attacks on ZKPs.

In this work we use ZKPs to prove that a ciphertext is an encryption of either of two plaintexts from Damgård and Jurik [DJ01] and that a vector of ciphertext is a shuffle of another vector of ciphertexts [Gro10].

# 4. Approximate String Matching

Parts of this chapter were published in [BK13].

## 4.1. Introduction

Matching strings and in general character sequences is a wide field and so generic that it is used in a large number of systems. Comparing strings privately is a specialization inside the security community with main use cases in the private record linkage and genome matching domain. Examples from literature are discussed in Section 2.1.2.

The most important requirements that a string comparison scheme should possess to fit the use case of a user carrying a mobile, resource-constraint device that stores sensitive information like the sequenced genome are highlighted and argued in Section 1.2.

The most fitting proposals for our desired solution were reviewed and underlying techniques evaluated against the requirements. None of the related work combined the important points of an efficient, privacy-preserving comparison scheme for long sequences with the ability to have inference control and supporting resource-constraint clients. Details on the related work can be found in Section 2.1.2.

Therefore we will construct (Section 4.2) the basic, efficient string comparison scheme, evaluate its security in Section 4.3 and measure its performance in Section 4.4. A quick interlude in Section 4.5 will discuss a generalization of the presented scheme towards approximating other measures. Finally, Section 4.6 concludes the construction of the basic comparison scheme, which is more efficient than the related work and already offers some basic *output* inference control.

In this chapter we will construct and evaluate the basic, efficient string matching scheme that forms the basis for all further constructions. The scheme will be extended in Chapter 5 to detect and mitigate inference attacks and further on in Chapter 6 to be resource efficient regarding encryption and transmission — while at the same time adding only minimal overhead on the server side.

## 4.2. Design

Two parties $A$ (Alice) and $B$ (Bob) have private input strings that should be compared. The desired comparison function is the edit distance between both inputs. The output is given to the initiating party (in our case Alice). However, to speed up the computation (which has a textbook time complexity of $\mathcal{O}(n \cdot m)$, with $n, m$ being the lengths of the input strings) the edit distance is approximated by an embedding into the Hamming distance over bit strings.

Alice could be the person holding her sequenced genome, or the difference of her genome to a reference genome. Bob could be a database owner belonging to a hospital or another health institution. Bob's task is to calculate the similarities between Alice's genomic sequence and the instances in his database, which might be reference values for certain classes of predispositions. The construction of the string comparison scheme follows the introduced preliminaries in Chapter 3.

The overall protocol will work like this:

1. Alice and Bob both convert their input strings $s_A, s_B$ into a set of variable length grams following the VGRAM algorithm [LWY07] described in Section 3.3.

2. Further, both map their set of variable length grams to a Bloom filter $b_A, b_B$, as described in Section 3.4.

3. Alice encrypts every bit of the Bloom filter $b_A$ using an Additive Homomorphic Encryption (AHE) scheme and sends the resulting vector to Bob.

4. Bob computes the exclusive OR inside the homomorphic encryption as shown in Figure 4.1.

5. Further, Bob homomorphically sums up all obtained results and sends the sum back to Alice.

6. Alice decrypts the result and finds the approximated distance.

Alice should learn the result of a comparison, while Bob should learn nothing. To achieve this we use the protocol depicted in Figure 4.1, which follows the description given above. The input for each party is its respective private string $s_A$ for Alice and $s_B$ for Bob. For ease of presentation we use $b[i]$ to identify the $i$-th element of array $b$. The index $j \in [1, l]$ in Figure 4.1 always runs over the whole length of the indexed array of length $l$, while $i \in [1, |g_A|]$ and $i' \in [1, |g_B|]$ runs over all generated grams.

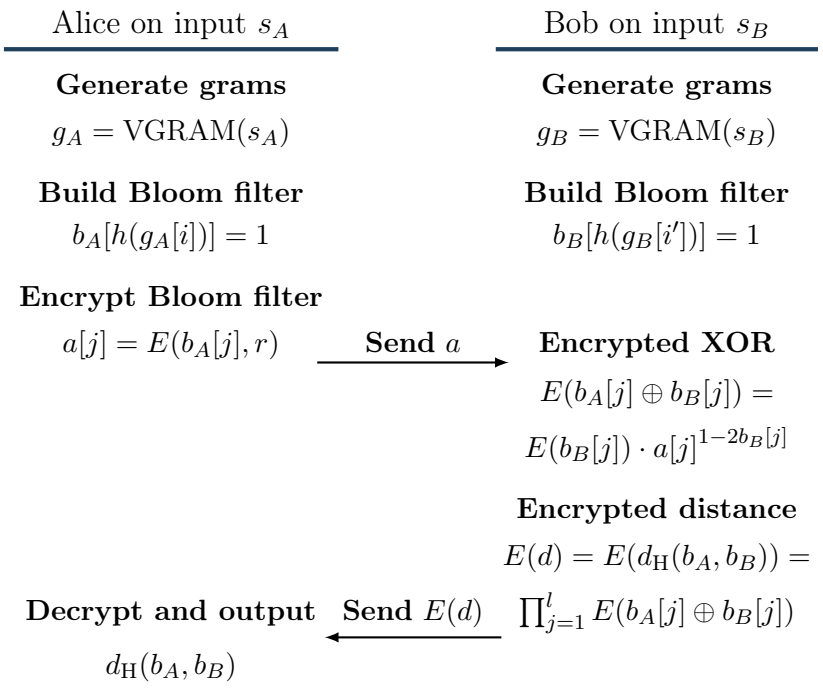



Figure 4.1.: Protocol for privacy-preserving distance measure using Bloom filters and homomorphic encryption. Alice and Bob input their genomic strings $s_A$ and $s_B$. Alice outputs a privacy-preservingly calculated set cardinality estimation as distance measure.

### Extension to Decrease Inference

Figure 4.1 specifies, that the calculated approximate distance value $E(d)$ between both compared strings $s_A$ and $s_B$ is returned to Alice for decryption so that she learns the actual distance value. This would however result in increased sensitivity to inference attacks, as they are described by Goodrich [Goo09]. To present an easy circumvention against such attacks, we restrict the information Alice gains from executing this protocol. Instead of learning the exact result of the comparison, the result is manipulated to give Alice only the information whether the distance is within a previously defined range $d \in [t_\mathrm{min}, t_\mathrm{max}]$ or not.

Following the classification from Section 2.3, this extended scheme represents an "output inference control scheme". We utilize removal and minimization of information, similar to techniques from privacy-preserving data-publishing. Other possible techniques are perturbation as done by differential privacy [Dwo08] or aggregation. In contrast to Chapter 5, the scheme presented there belongs to the "input inference control schemes", similar to actively preserving privacy in statistical databases through identification of inference attack queries.

Recall that the calculated distance value $d$ equals the Hamming distance between the Bloom filters $b_A$ and $b_B$. Furthermore, Li, Wang, and Yang [LWY07] present in Section 5.3, Lemma 1, equation (4) an upper bound for the Hamming distance between bit vectors of gram sets generated using their VGRAM algorithm. The actual calculation of the upper bound involves the number of affected grams for both input strings $s_A$ and $s_B$. As $s_B$ is not available to Alice, she uses the revised Cambridge Reference Sequence (rCRS) [And+81] $s_{\text{rCRS}}$ as a reference to replace $s_B$ in the calculation of the upper Hamming distance bound. This replacement is a good approximation for small edit distances due to a high similarity between the rCRS and other mitochondrial genomic sequences. Following [LWY07] the upper bound $t_{\max}$ for a maximum edit distance $d_{\text{E}_{\max}}$ is calculated as $t_{\max} = \text{NAG}(s_A, d_{\text{E}_{\max}}) + \text{NAG}(s_{\text{rCRS}}, d_{\text{E}_{\max}})$, where $\text{NAG}(s, d_{\text{E}})$ describes the maximum number of affected grams for $d_{\text{E}}$ edit operations on string $s$. Further, as $\text{NAG}(s_{\text{rCRS}}, d_{\text{E}_{\max}})$ is very close to $\text{NAG}(s_B, d_{\text{E}_{\max}})$ it is thus used as an approximation. $\text{NAG}(s_A, d_{\text{E}_{\max}})$ can also be replaced for the same reason. It follows that the approximation for $t_{\max}$ does not depend on the input sequences $s_A$ and $s_B$ and can be determined before running the protocol. The estimation of $t_{\min}$ for $d_{\text{E}_{\min}}$ is performed equivalently. These thresholds can later on be used to decide if the edit distance between input sequences $s_A$ and $s_b$ is above a certain edit distance threshold $d_{\text{E}_{\max}}$. This functionality will be used in the extended protocol described below.

Due to the probabilistic nature of the Bloom filter, elements are mapped to the same positions with a probability $p_{\text{fp}}$ as described in Section 3.4. As the expected Bloom filter cardinality $|b_A| = \sum_{i=0}^{l} b_A[i]$ for a Bloom filter $b_A$ of length $l$ is therefore smaller than the number of inserted elements — or in our case variable grams for $s_A$ — the upper bound must be corrected to take the Bloom filter hash collisions into account.

**Cardinality Analysis** Swamidass and Baldi [SB07] analyzed for a given Bloom filter of length $l$, cardinality $i'$ and $k$ hash functions what the most likely number of originally inserted values $i$ is. They showed that $i = -l/k \cdot \ln(1 - i'/l)$ is a very good approximation of the probable number of inserted elements. As discussed in Section 3.4, we use $k = 1$ hash functions, which is optimal, and can therefore simplify the equation to $i = -l \cdot \ln(1 - i'/l)$. Once this equation is transposed to estimate $i'$, it defines the most probable number of bits set after inserting $i$ elements into a Bloom filter of length $l$.

$$i' = \left(1 - \exp\left(-\frac{i}{l}\right)\right) \cdot l \tag{4.1}$$

This cardinality estimation from Equation (4.1) is then used for calculating the corrected threshold values $t_{\min}$ and $t_{\max}$ as follows.

$$t_{\min} = \left(1 - \exp\left(-\frac{\mathrm{NAG}\big(s_{\mathrm{rCRS}}, d_{\mathrm{E}_{\min}}\big)}{l}\right)\right) \cdot l \tag{4.2}$$

$$t_{\max} = \left(1 - \exp\left(-\frac{\mathrm{NAG}(s_{\mathrm{rCRS}}, d_{\mathrm{E}_{\max}})}{l}\right)\right) \cdot l \tag{4.3}$$

Using a Homomorphic Encryption (HE) scheme, as described in Section 3.5, the generated thresholds can be encrypted and used for further processing. Recall that $E(t_i, r)^{-1}$ denotes the multiplicative inverse element of the encryption $E(t_i, r)$, found through executing the extended euclidean algorithm, which is by the homomorphism definition the encryption of the additive inverse plaintext.

$$E(t_i, r)^{-1} = E\big(-t_i, r'\big)$$

By using the extended euclidean algorithm to find a homomorphic additive inverse and applying a homomorphic addition afterwards, the homomorphic subtraction can be defined, as it was described in Section 3.6.

**Extended protocol** To extend the basic protocol depicted in Figure 4.1, the server Bob has to perform a few more steps before the result is sent to Alice. The extension fits in the original protocol just after Bob calculated the encrypted distance and thus just before he sends the encrypted distance to Alice. The extended protocol is depicted in Figure 4.2 and replaces the last step of the original protocol, which was sending the distance to Alice and final decryption of the encrypted distance value. Calculating the return values for Alice by Bob for the output inference control extension works as follows.

The algorithm generates all possible integer values within the threshold range $[t_{\min}, t_{\max}]$ and subtracts the calculated distance $d_{\mathrm{H}}(b_A, b_B)$ from every value in this range. Observe that if $t_{\min} \leq d_{\mathrm{H}}(b_A, b_B) \leq t_{\max}$, there will be an encryption of the zero element in the resulting vector of all differences. This difference vector will then be multiplied with random factors and randomly permuted. The resulting vector is then sent to Alice for complete decryption. If Alice finds a zero after decryption she knows that the distance was within the threshold range, otherwise it was not. Section 4.3 will recall the steps and show for each that privacy is preserved and what information is leaked (or not).

The protocol can be split up in three parts. The first part would be the mapping of inputs to Bloom filters on both sides, including the encryption and transmission done by Alice. The second part spans all remaining computations that Bob is performing including the transmission of results back to Alice. The last (third) part then includes the necessary steps (decryption and zero check) on Alice side to get to the result.

The extended protocol is identical to the basic protocol from Figure 4.1 until the red dashed line, then — instead of just sending the encrypted distance back to Alice — further steps on Bob's side are applied. Section 4.3 will discuss these steps in detail regarding their privacy implications. We are going to look at the additional steps regarding their functionality.

Recall that the goal of the extended protocol is to allow Alice to learn only whether the distance between the input strings is within a predefined range or not. We use an approximation of the edit distance as the distance measure. So after calculating the encrypted distance between both strings, the idea is that Bob generates all possible values within the threshold range $[t_{\min}, t_{\max}]$, which means $o = t_{\max} - t_{\min} + 1$ consecutive integer values. Further he subtracts them all from the actual encrypted distance $E(d - t_i)$ to derive an array $d'$ of size $o$. Let's seperate the two cases that Alice should be able to separate.

First, if the actual distance $d$ between both strings is not within the threshold range, that is either $d > t_{\max}$ or $d < t_{\min}$, then all values in $d'$ are either above 0 iff $d > t_{\max}$ or below 0 otherwise. However, if $d$ is within the threshold range, then there is a single value in $d'$, which is zero. This is exactly the test that Alice performs once she receives the encrypted array. She checks whether one of the decrypted values is zero. If there is a zero, she knows that the string distance is within the threshold range, otherwise she learns that the distance is outside the checked range. All the other steps, which were not mentioned are to remove unnecessary information that Alice could use to infer the actual distance.

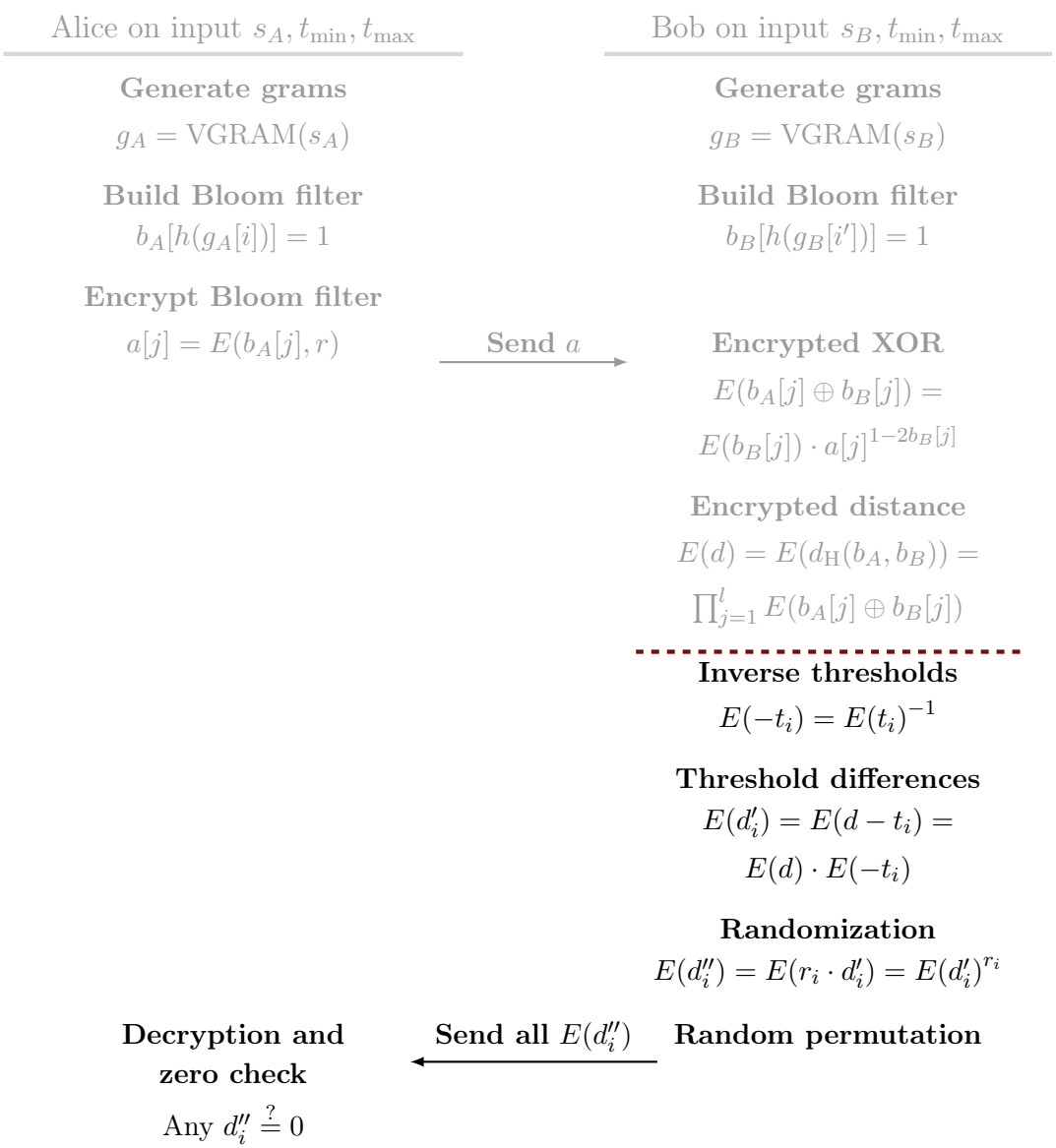



Figure 4.2.: Extended protocol for privacy-preserving, decreased-inference distance measure using Bloom filters and homomorphic encryption. Alice and Bob input their genomic strings $s_A$ and $s_B$. Both previously agreed upon a threshold range $[t_{\min}, t_{\max}]$ to test the distance against. Alice outputs a binary answer whether the calculated distance measure is within a predefined threshold. The gray part is identical to the basic scheme depicted in Figure 4.1.

## 4.3. Security Analysis

The security properties of the described protocols are implications of the underlying homomorphic cryptosystem, *e.g.* Naccache-Stern [NS98]. Exchanging the homomorphic system thus also affects the security properties. Typically the AHE system guarantees semantic security, Indistinguishability under Chosen Plaintext Attack (IND-CPA) and thus input privacy against a Honest-But-Curious (HBC) attacker, not pursuing an inference attack. Furthermore, our protocol guarantees input privacy for Alice even in case of an malicious server Bob, as Bob never receives any data that depends on input from both parties. Furthermore, Bob does not receive any output and can thus not perform an inference attack. The only data that Bob receives is the encrypted input from Alice, which is guaranteed to be semantically secure by the implemented crypto system.

The only influence that Bob can exert is in manipulating the choice of the threshold range $[t_{\min}, t_{\max}]$. The selection of these values can have an effect on how much information can be inferred from the results of the protocol. Both parties must ensure that viable and non-intruding thresholds are chosen. The choice of these thresholds to achieve a good utility-privacy trade-off are out-of-scope for this thesis.

**Protocol Dissection** In the first part of our protocol, Alice translates her input string into variable length grams, generates a Bloom filter representation and encrypts it using a public key cryptosystem. As she is not using any information from Bob, she cannot gain any insight into Bob's input.

The second part involves Bob working on the encrypted Bloom filter from Alice and her encrypted Bloom filter cardinality. As all values are encrypted using an asymmetric, probabilistic cryptosystem that is semantically secure and for which only Alice has the private decryption key, Bob cannot decide if an encrypted value represents a zero, a one or any other value, which directly follows the guarantees of the underlying hardness assumption. The number of elements received does not depend on Alice input, as only public information is used to set the Bloom filter length, as introduced in Section 4.2. Further Bob sums up elements from Alice's encrypted Bloom filter, based on his Bloom filter. The result is then subtracted several times from different public threshold values and multiplied with random numbers, chosen uniformly from within the domain of plaintexts of the underlying cryptosystem. All results are shuffled at random and transmitted back to Alice. Bob gained no information in this phase about Alice's input.

As a last step Alice decrypts all results received from Bob and checks if they contain a zero. If a zero is found, she learns that the Hamming distance between

the Bloom filters was within a predefined threshold $[t_{\min}, t_{\max}]$. There can be at most one zero. If no zero was found, the Hamming distance was outside the range. From the decrypted non-zero results, she cannot learn anything, as these numbers are uniformly distributed due to the multiplication with uniform random numbers drawn from the plaintext domain modulo the plaintext domain modulus. The index of the zero element, if there was one, gives no information to Alice, as the return values were randomly shuffled by Bob. The number of returned elements also holds no further information, as there are always $t_{\max} - t_{\min} + 1$ results if no multiplicative homomorphic system is used, or fewer if the homomorphic cryptosystem supports multiplications and these multiplications are used for aggregation of results. Even if the returned array is aggregated to some arbitrary degree, this does not use any information from the private input, but solely depends on the multiplicative properties of the underlying homomorphic system and the configuration on how many multiplications to use for aggregation.

The only information Alice learns about the input of Bob is, if the Hamming distance towards his Bloom filter $b_B$ lies within the threshold range.

## 4.4. Evaluation

We are going to evaluate the effectiveness of the proposed scheme by looking at the correlation between the original edit distance and the approximation given by the Hamming distance. Furthermore, a mapping from the Hamming distance to a possible edit distance is given and an extension to increase accuracy is proposed. A performance evaluation shows that the algorithm is performance-wise at the same level as comparable algorithms with similar privacy-preservation but less functionality, as explained in Chapter 2.

For the experiments a Linux Laptop with an Intel Core2 Duo T9600 running at 2.8 GHz with 4 GB RAM was used. The code is written in Java, using the Bouncy Castle library[1]. The first tests evaluate the relation between the Levenshtein distance and the Hamming distance as introduced in Section 4.2. Further, the runtime performance of the algorithm is evaluated for string lengths also used in the literature for comparable privacy-preserving string matching protocols.

### 4.4.1. Distance Measure

As our distance measure approximates the Levenshtein distance, we measure the relation between the edit distance and the Hamming distance over Bloom filter.

[1] http://www.bouncycastle.org/

Li, Wang, and Yang [LWY07] already specifies a maximum Hamming distance for a certain edit distance given bit encodings of the equivalent variable length gram sets. However, our binary strings have slightly different properties. First of all we cannot use the NAG vector function upon the private inputs, but estimate their results using the reference sequence $\mathrm{NAG}(s_{\mathrm{rCRS}}, e)$ instead. Second, our mapping to bit strings is probabilistic, producing strings with hamming weight smaller than the number of variable grams. Therefore we combine these theoretical upper limits with the approximation of the final Bloom filter cardinality from Equation (4.1), introduced by Swamidass and Baldi [SB07].

The first practical issues arise for selecting useful values for the minimum and maximum gram length that is used in the VGRAM algorithm. The parameters are denoted $q_{\min}$ and $q_{\max}$ equivalently. Luckily Li, Wang, and Yang [LWY07] proposes to start with a rather small $q_{\min}$ and a large $q_{\max}$. Following this approach the parameters $q_{\min} = 2$ and $q_{\max} = 40$ are used. Appropriate values can later be extracted during the execution of their internal data structure optimization (pruning the built Trie). The lower value was set to 2 as we operate on a 4 character alphabet for genomic sequences and having a single character without its position carries nearly no information about the original sequence. In this sense a bi-gram is probably also much too short, but the algorithm must start somewhere and can handle too small gram lengths. The upper value is a practical limit as the test machine was not able to handle larger gram lengths due to its limited main memory. However, it seems that using smaller values doesn't influence the results much.

To run the tests the Bloom filters are set up to use a single hash function as discussed before. In our case "SHA1" is used to calculate a hash of an element. The hash is reduced modulo the size of the filter, except for when the largest integral multiple of the filter length — which is still smaller or equal than the maximum value returned by the hash — is smaller than the actual hash value. In such a case the hash function is called again with a non-alphabet character and an incremental number appended to the actual element.

The mitochondrial genomic sequences contain roughly $n = 16569$ characters, which means that about the same number of variable grams had to be inserted into each Bloom filter. The probability that a false-positive test occurs after the $n$ elements are added to the filter was set to $p_{\mathrm{fp}} = 0.1$ and $p_{\mathrm{fp}} = 0.5$ to either keep the error introduced by false-positives of the Bloom filter rather low, or support the upcoming requirements from the Error-Correcting Code (ECC) in Chapter 5. Following equation (3.2) these parameters required a Bloom filter of size $l = 157261$ bits or 23905 bits respectively. The influence of the false-positive rate setting on accuracy will be discussed in more detail in Section 5.7.

To perform the runtime analysis we chose random strings from the human mitochondrial DNA database [IG06] and applied $e$ edit operations to it, such that in the

end there are always two strings with edit distance $e$. Edit operations were chosen so that they do not cancel themselves out or produce shortcuts which evaluate to lower edit distances. Each edit distance $e$ was measured 100 times for repeated runs for all $e$ in the range $[1, 100]$. Thus overall 10000 randomized comparisons were made. The chosen edit distance range is well fitting, as the average human mitochondrial genome pairwise substitution distance is 28 with a standard deviation of 18 due to Goodrich [Goo09], our tests thus include the majority of cases that will happen when comparing two full human mitochondrial genomes.

After choosing a fixed number of edit operations, they are used to create pairs of private inputs $s_A$ and $s_B$. Subsequently the extended protocol depicted in Figure 4.2 is run. The original string and the altered string are thus compared using our distance measure. The resulting distance value $d$ is the difference between the union cardinality and the intersection cardinality of both Bloom filters $b_A$ and $b_B$. This represents the total number of unique elements for both parties, or the Hamming distance between both Bloom filters. Following Lemma 1 in [LWY07] this directly correlates with the Levenshtein distance between the strings.

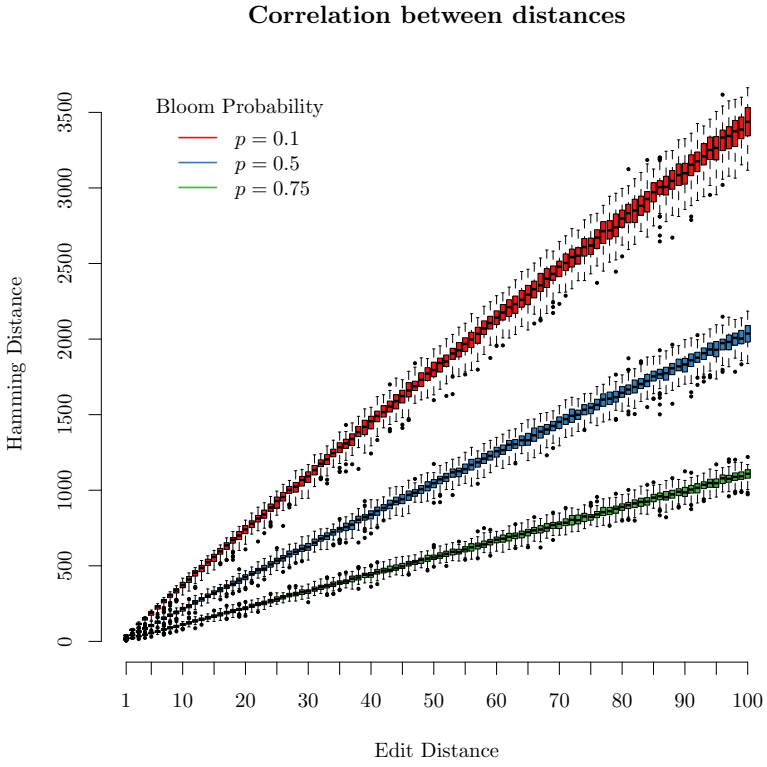



Figure 4.3.: Relation between the original edit distance of two strings and the Hamming distance between the according Bloom filters.

Figure 4.3 shows a Boxplot for every Levenshtein distance in the range $[1, 100]$ depicted on the $x$-axis and calculated over 100 runs. As can be seen from the figure, our distance value approximates and especially separates small Levenshtein distances very good, with a narrow range of possible Hamming distance values and a small variance. The Pearson correlation between the Levenshtein distances and the Hamming distances between the corresponding Bloom filters is $c_p = 0.997$ for up to 100 edit operations.

We can also see that the variance in our approximated distance grows with larger edit distances and that the means are not growing linear, but sub-linear, which together leads to higher overlaps of Hamming distance ranges for consecutive edit distances. Figure 4.4 gives an idea for the distributions. As a result, exact reconstruction of the original edit distance is not possible. However, the Hamming distance distributions overlap only slightly for small edit distances. For higher distances over the inputs, probabilities for a range of possible edit distances can be given. This means that approximations for larger Levenshtein distances are less accurate than for smaller ones.

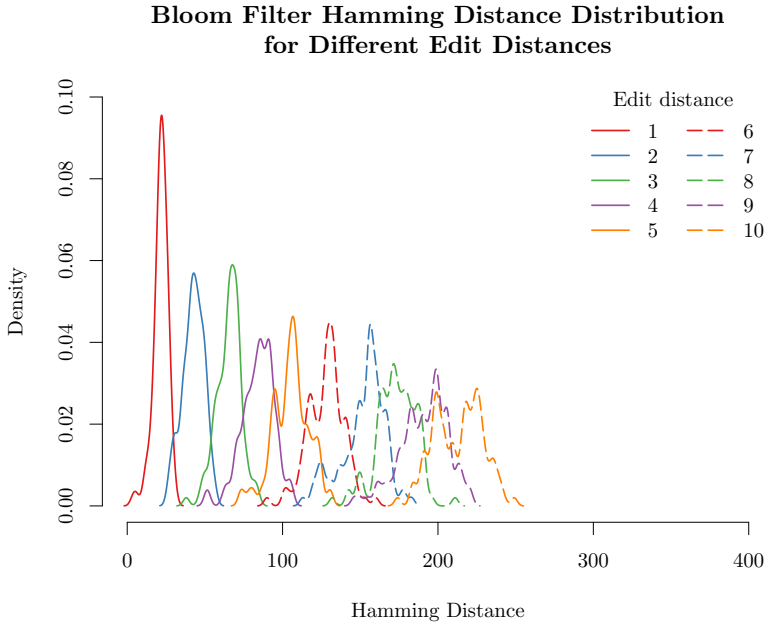



Figure 4.4.: Distribution of Hamming distances between Bloom filters for different edit distances between the original strings $s_1, s_2$. The Bloom filters were constructed using a false-positive probability of $p_{\text{fp}} = 0.5$.

Figure 4.4 gives several distributions of Hamming distances over Bloom filters generated by the protocol described in this chapter. Each plotted distribution corresponds to results obtained for a different edit distance between the input strings.

The Bloom filter was further generated to have a final false-positive probability of $p_{\text{fp}} = 0.5$. One important task is to have a good mapping between the edit distance and the Hamming distance upon Bloom filters, as this reduces false-positives and false-negatives in the extended protocol. Distributions for small edit distances can be separated well, while distributions overlap more for higher edit distances. As an example: deciding between an edit distance 1, 5 or 10 was possible in all cases based on the Bloom filter Hamming distance, which can already be an important use case for distinguishing between different types of genetic mutations. Point mutations typically affect a single base pair, causing an edit distance of one for the sequenced section of the genome, while a single repeated expansion may affect a few base pairs and DNA duplication possibly yields a very high edit distance to a large number of insertions.[2]

### 4.4.2. Protocol Execution Time

To evaluate the performance of our protocol, we used the same configuration as above, that is 100 runs for each edit distance test. The parameters were thus set to $q_{\text{min}} = 2$, $q_{\text{max}} = 40$, $p_{\text{fp}} = 0.1$ and actual runtimes for a maximum edit distances of up to 10 operations ($t_{\text{max}}$ is set using an $d_{\text{Emax}} = 10$ in Equation 4.3) are shown in the following figures.

The client runtime depends linearly on the length of the input sequence, where the most time is spent on encrypting the Bloom filter prior to transmission and decrypting the results from the server. We can see a pretty high variance on client runtimes, with a linearly growing mean over longer sequences (see Figure 4.5). This behavior is easily explained by the unknown number of results which must to be decrypted until a zero is found. If the distance between both compared sequences is not within the predefined range given by the thresholds, the client always needs to decrypt all results, as no zero will be found within the returned values. Thus the maximum time is used. On the opposite, if the distance is in range, then the time spent on decrpting the results might be considerably less, as the client stops once a zero is found. This might introduce a side-channel on timing if there are subsequent protocol steps after the client side decryption of results.

Server runtime depends linearly on the threshold range size, whereas runtimes for different sequence lengths are only increasing slightly. The measured values for a constant threshold derived from a maximum edit distance of 10 and a variable sequence length range between 7.45 seconds for sequences of length 200 and 7.8 seconds for full mitochondrial DNA sequences. Empirical results are depicted in Figure 4.6.

[2]Possible genetic mutations and effects upon base-pairs taken from the U.S. National Library of
Medicine `http://ghr.nlm.nih.gov/handbook/mutationsanddisorders/possiblemutations`

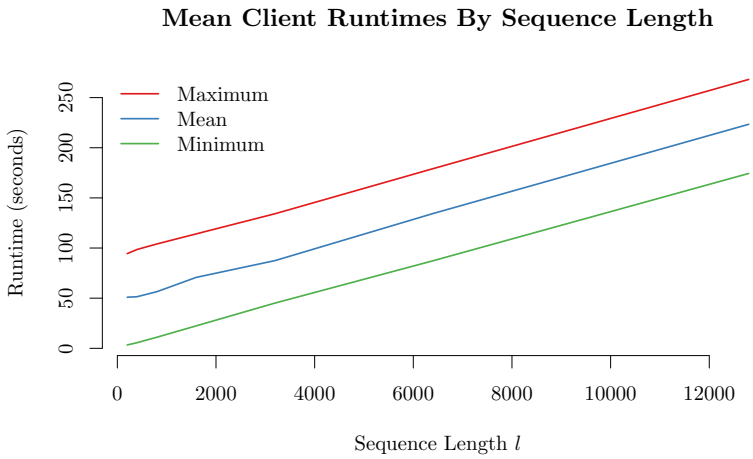



Figure 4.5.: Overall runtime for different sequence lengths at client side. Required resources grow linear in the sequence length. The gap between the minima and maxima is the time necessary for decrypting all results from the extended protocol. The mean is calculated for comparisons which returned *true* and thus had a distance within the threshold range. Non-similar results always need maximum time.

The amount of data that needs to be transferred between Alice and Bob is shown in Table 4.1 and grows linearly with the length of the Bloom filter for the traffic from Alice to Bob and linearly with the size of the threshold range for the traffic from Bob to Alice. For this test the threshold $t_{\max}$ is set to the maximum Hamming distance defined according to Equation 4.3 in Section 4.2 for a maximum edit distance of 10, $t_{\min}$ is set to 0 to match all strings up to a certain edit distance. An edit distance of 10 is used as the majority of the results can be correctly mapped back to the original edit distance — the range between the first and third quartile of the Hamming distance distribution for a single edit distance does not overlap with the first to third quartile range of the next lower and higher edit distance. This separation of ranges can be seen in Figure 4.3.

Looking at the results given by Jha, Kruger, and Shmatikov [JKS08] and Huang, Evans, and Katz [HEK11] in the evaluations of their state of the art protocols, we achieve a similar performance with a low constant crypto overhead starting with the smallest sequence lengths of 200 characters. Due to the lower linear complexity of our protocol it scales better (we have a lower multiplicative factor) and thus comparisons of full mitochondrial DNA sequences can be performed more efficiently.

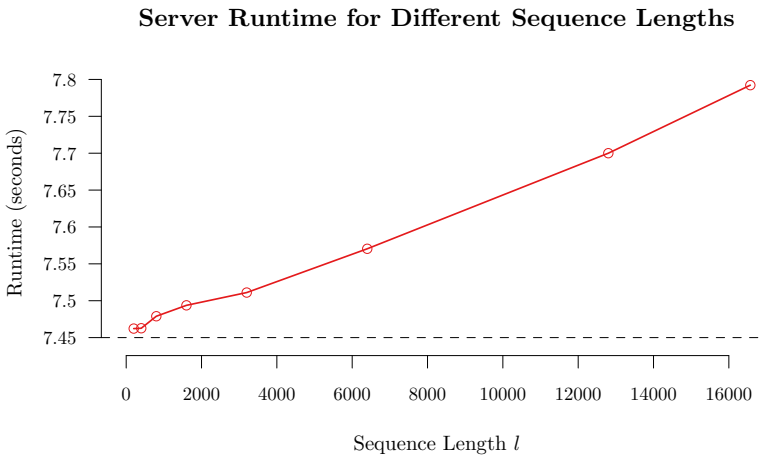



Figure 4.6.: Overall runtime for different sequence lengths at server side. Required resources grow linear in the sequence length.

| Sequence Length $l$ | Client $\rightarrow$ Server | Server $\rightarrow$ Client |
|---:|---:|---:|
| 200 | 296 KB | 123 KB |
| 400 | 590 KB | 123 KB |
| 800 | 1169 KB | 123 KB |
| 1600 | 2337 KB | 123 KB |
| 3200 | 4663 KB | 123 KB |
| 6400 | 9323 KB | 123 KB |
| 12800 | 18636 KB | 123 KB |

Table 4.1.: Size of transmission for different sequence lengths and both directions.

## 4.5. Extensions

In case the homomorphic cryptosystem also supports at least one homomorphic multiplication — as is the case with the BGN cryptosystem presented by Boneh, Goh, and Nissim [BGN05] — the computation could be completely outsourced to a HBC third party. But as this is not fitting our requirements as described in Section 1.2, we will not dive deeper into outsourcing computations.

Further, with homomorphic multiplications at hand the number of returned elements by Bob can be reduced very effectively. Specifically if the underlying homomorphic encryption system allows at least $\lceil \log_2(t_{\max} - t_{\min} + 1) \rceil$ multiplications, Bob can reduce the array $d''$ to a single element. For the BGN system [BGN05], which allows a single multiplication, the number of array elements can be reduced

from $o$ to $\lceil o/2 \rceil$ by taking any two values which were not multiplied before and multiply them. This way, multiplying two values which are not zero will never result in a zero, as zero is not part of the multiplicative group. Furthermore, multiplying zero with any other value will always return a zero. This way if there was a zero in the original array $d''$, there will be a zero in the aggregated array, too. If there was no zero element in $d''$ there will be none in the aggregation.

If the system supports two multiplications, Bob takes the resulting $\lceil o/2 \rceil$ elements from the first aggregation turn and multiplies pairs of elements which have been multiplied up to once. The result is an array of size $\lceil o/4 \rceil$. This way a reduction of the size of the vector returned from Bob to Alice, exponential in the number of multiplications, can be achieved. A possible homomorphic cryptographic system is a BGN-style system, which uses multi-linear maps instead of bilinear maps, as described by Coron, Lepoint, and Tibouchi [CLT13]. Other possible systems are those presented by Doroz, Hu, and Sunar [DHS14], Lepoint, Coron, and Tibouchi [LCT14], Cheon et al. [Che+13], and Lepoint and Naehrig [LN14].

**Improving approximation accuracy** In the proposed scheme an approximation of the original edit distance $d_{\mathrm{E}_o}$ can be performed by looking up the most probable edit distance that could produce the resulting hamming distance $d_{\mathrm{H}}$. This lookup can be performed by constructing and evaluating probability functions $p_i,\ i \in [1, d_{\mathrm{E}\max}]$ that specify the probability of a Hamming distance for a certain edit distance. $d_{\mathrm{E}\max}$ is the maximum edit distance $d_{\mathrm{E}_o}$ being considered. Figure 4.4 shows plots of such probability functions generated by measurements. The actual approximated distance $d'_{\mathrm{E}_o}$ can then be selected by $d'_{\mathrm{E}_o} = \{i | \max(p_i(d_{\mathrm{H}})),\ i \in [1, d_{\mathrm{E}\max}]\}$.

This approximation can be improved in case the approximated metric has some invariance property or is homogeneous linear. Examples include transposition-invariant string metrics, translation-invariant distances between points or the invariance of angles and for rotations. The edit distance is limited transposition-invariant [MNU05], which is enough in our case. Improving the approximation then works by running the protocol several times given randomly translated, transposed or otherwise randomized input regarding the invariance property. It is known that the approximation for all results must be the same, so a simple majority vote over the results improves the approximation.

The protocol described in Section 4.2 is used in the genome sequence matching setting, but can of course also be used for arbitrary other sequences of elements over which either the edit distance can be adopted, or the Hamming distance itself can be defined. Of course the protocol can also be used in a broader scope and for different measures if a low distortion embedding into a supported set measure or the string edit distance can be found. Ostrovsky and Rabani [OR07], Akutsu

[Aku06], Akutsu, Fukagawa, and Takasu [AFT06], and Bille [Bil05] describe such embeddings.

Furthermore, we also support the exact evaluation of the Manhattan distance or $l_1$-norm without any approximation by replacing our embedding of a string via the VGRAM algorithm and the subsequent Bloom filter with the actual values (vectors) from the $l_1$-space. This way approximations for other metrics for which embeddings into the $l_1$-space are known can be efficiently evaluated. Indyk and Matousek [IM04] give an overview on known embeddings into $l_p$ and further Bourgain [Bou85] describes generic embeddings from any $l_p$-space into $l_1$ together with the expected distortion.

## 4.6. Conclusion

We presented a novel, non-interactive approach for a privacy-preserving approximate string matching protocol, that achieves higher performance than comparable state-of-the-art proposals and can scale to real-world sized mitochondrial genomes. A user who tries to attack the server via inference queries will not even learn the exact distances or approximations, but only whether two compared strings are within a predefined distance range — if the extended protocol was used.

Due to the computational complexity being linear in the length of the longest input sequence: $\mathcal{O}(\max(|s_A|, |s_B|))$ and the communication complexity being linear in the size of the threshold range $o$, respectively in the Bloom filter length $l$: $\mathcal{O}(\max(o, l))$, this protocol is very practical and was tested for full mitochondrial sequences with roughly 16500 base pairs, which are encoded as the same amount of characters. A sequence comparison took about 286 seconds on the mentioned hardware to complete.

As the most time is spent for encryption, transmission and decryption, techniques as presented in Chapter 6 together with a fitting threshold homomorphic encryption system [CDN01; DJ01] are apt to reduce the absolute wall time from the start till the successful completion of the comparison protocol.

The efficiency can also be increased by trading matching accuracy against performance by using a sketching algorithm as described by Bar-Yossef et al. [Bar+04]. They find different embeddings from the edit distance into Hamming space. They especially find an embedding from an arbitrary string into a Hamming space with constant dimensions to decide the gap-Levenshtein problem. This is the problem of deciding whether the distance between both strings is below $k$ or above $l$.

Chapter 5 will show that also Bloom filters with much higher false-positive rate can still be used for accurate edit distance estimation and thus instantly have a much decreased runtime.

# 5. Inference Control

Parts of this chapter were published in [KBS14].

## 5.1. Introduction

Following the overall goal of building an efficient, privacy-preserving comparison scheme, we designed a basic comparison scheme in Chapter 4, instantiated it as an approximate string matching scheme in Section 4.2 and described its generalization in Section 4.5. Its security, effectiveness and efficiency was thoroughly analyzed. Recalling the identified open research questions from Section 1.3, as well as the scheme requirements outlined in Section 1.2, emphasis will be put on inference control to reduce the efficiency of inference attacks. Up until now a privacy-preserving string matching scheme was designed — together with a framework for privacy-preserving comparisons —, which can minimize the amount of information returned. However, the presented in-range-check might not be applicable in all cases, particularly when more information than a boolean range check is necessary.

Nevertheless, Goodrich has shown that even in the case of a privacy-preserving protocol significant information may often be leaked [Goo09]. Using only the result of the comparison, the querying party Alice can guess Bob's string by repetitive invocations. Surprisingly few queries are actually necessary to infer even long, real-world sized genomes, *e.g.* for a sample of 1000 human mitochondrial genomic sequences Goodrich's attack inferred 90% of them using 875 queries or less.

In this chapter we will design a privacy-preserving protocol that also can detect (and mitigate) whether an inference attack is taking place. If no attack is taking place, *i.e.*, in the case of an honest Alice, both parties' privacy will be protected and Alice will learn the result of the protocol. If an attack is taking place, Alice will not receive the answer to "suspicious" queries. We use Goodrich's attack as a template, but we detect and mitigate all types of attacks that leverage close queries in terms of the hamming distance over pairs of queries.

We validate against Goodrich's attack, since it is well known and designed to be particularly efficient in this type, but our approach is more general. We construct a

generalization of Goodrich's attack and use it for evaluation. An interesting result is that the structure of queries artificially generated for inferring information via close queries and the typical mutations that happen on genomes can be separated quite well for simple attack schemes. This again greatly helps reducing false-positives and false-negatives for inference attack detection in Goodrich-like attacks.

Inference attack detection on encrypted data is a complicated, not straightforward approach. First, Bob should not know Alice's query string in order to preserve privacy. It must remain encrypted. Therefore the inference control algorithm must also work on encrypted data; to the best of our knowledge we are the first to develop and analyze such an inference control algorithm. This could still be implemented using a generic secure computation, but, second, all inference control algorithms work on the entire history of Alice and match the query against all previous queries by Alice [Dom08]. Therefore the secure computation must span the entire history of queries, quickly making it impractically inefficient with the inherent quasi-quadratic complexity in the number of inputs $\mathcal{O}(n^2)$.

Instead, in our solution the cryptographic operations are limited to a single query and inference control can be performed using simple (plaintext) equality matching over the history, denoted $C_{\text{hist}}$. Yet, we need to trade some precision of the inference control mechanism, *e.g.*, compared to logic-based methods, for this efficiency while still effectively mitigating the attack.

Our construction achieves this by using two different primitives and works as follows: We use the secure computation genome matching protocol based on Bloom filters and homomorphic encryption described in Chapter 4. In addition to the encrypted Bloom filter Alice submits a fuzzy (but deterministic) commitment [JW99] of the Bloom filter using an Error-Correcting Code (ECC). Bob can match those fuzzy commitments and will not answer the query if Alice has committed to a similar value before, such that "close" queries are prevented. The idea is that by only answering queries from Alice, which are not close to all previous queries, an uncertainty about Bob's string remains and thus the efficiency of inferring information is greatly reduced. We are going to evaluate the effect of this uncertainty upon the inference attack.

An obvious attack on the sketched scheme is that Alice could use different Bloom filters in secure computation and fuzzy commitment. So, an important contribution of this chapter is an efficient zero-knowledge proof — not resorting to generic constructions — that Alice actually used the same Bloom filter in both. Furthermore, Alice could compute the ECC incorrectly. For this, we also contribute a zero-knowledge proof that Alice submitted an information word.

We emphasize that our approach to mitigate Goodrich-like attacks is detective. Our algorithm groups close patterns to match an on-going attack. As such, our

inference control algorithm for mitigating attacks is probabilistic and thus may give false-positive and -negative results. This implies that there is no provable guarantee that an attacker cannot infer the genome in the database and furthermore, some legitimate queries may be rejected. On the contrary, our guarantee of privacy — and soundness in the zero-knowledge proof — is provable.

In summary, this chapter contributes

- a privacy-preserving comparison scheme that allows *detection of similar* ***encrypted*** *inputs*
- an efficient *zero-knowledge proof* that the client has submitted its input truthfully and does not evade detection
- an *analytical and emperical analysis* of our detection scheme and how it mitigates Goodrich-like attacks on privacy of genome matching.

The remainder of the section is structured as follows. Section 5.2 references related work specific to privacy-preserving genome matching and inference control. We introduce the building blocks used in our protocol in Section 5.3. In Section 5.4 we describe the overall design of the inference attack detection construction. We describe our zero-knowledge proof in Section 5.5 and its security proof in Section 5.6. Section 5.7 presents the analysis of our scheme under Goodrich's and similar attacks. Furthermore, Section 5.8 describes an empirical analysis of the inference attack detection scheme under various parameters and demonstrates its applicability. Section 5.9 concludes the chapter on inference control.

## 5.2. Related Work

Privacy-preserving matching of genomes has been introduced in [AKD03]. It presents a secure computation based on homomorphic encryption, such that both the querier's genome and the database's genome are protected. The implemented string metric via secure computation is the edit distance. The original setup described in [AKD03] has been improved in performance using Yao's protocol [Yao86] for secure computations in [JKS08]. Although further improvements to Yao's protocol yield even better performance in this computation [HEK11] it is still too slow for large scale deployment.

Therefore different approaches to computing the edit distance were sought. Automata and regular expressions can emulate edit distance computations efficiently for small edit distances. An oblivious evaluation of automata is presented by Troncoso-Pastoriza, Katzenbeisser, and Celik [TKC07], but due to the regular expressions it does not scale to real-world sized genomes. Bloom filters can also be

used to estimate the edit distance. An evaluation of this approach using homomorphic encryption is presented in [BK13]. This approach yields reasonable runtimes (approx. 5 minutes) for real-world sized mitochondrial genomes (approx. 16568 characters) and we therefore build upon this approach.

In order to improve performance, simplified algorithms — compared to the edit distance — are considered. Baldi et al. [Bal+11] use simple set intersection to match genomes. Of course, the applicability to life sciences is limited. Instead, life sciences use increasingly complex algorithms, of which some have already been made privacy-preserving. An example are hidden Markov models, which are used to privacy-preservingly analyze gene sequences by Franz et al. [Fra+12].

Although secure computation offers a formal security model — semi-honest security [Gol04] (see Section 1.5.1), it does not prevent inferences from the result. Goodrich therefore presented an attack based on the information of the edit distance alone [Goo09]. With very few repeated queries it can infer real-world sized genomes. The contribution of this chapter is to combine the efficient protocol presented in Chapter 4 with a mitigation technique against this and similar attacks.

The guaranteed randomization approach to prevent inferences about the input is differential privacy [Dwo08]. A randomized function $K\colon D \to \mathbb{R}^d$ gives $\epsilon$-differential privacy if, for all data sets $D_1$ and $D_2$ differing on at most one element and all $S \subset \mathit{Range}(K)$,

$$\Pr[K(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[K(D_2) \in S].$$

It means, that the likelihood of any function result will only marginally change with the presence or absence of one additional element. Let $\Delta K$ denote the sensitivity of the function $K$ with $\Delta K = \max_{D_1,D_2} \|K(D_1) - K(D_2)\|_1$ and $\mathsf{Lap}(\lambda)$ be a random variable sampled from the Laplace distribution. The Laplace distribution is defined by the probability density function $\Pr[x|\lambda] = \frac{1}{2\lambda} \exp(-\frac{|x|}{\lambda})$. So the parameter $\lambda$ directly controls the magnitude of the random variable $\mathsf{Lap}(\lambda)$, which is called noise.

Dwork [Dwo08] proposed to perturb output $K(D)$ using noise sampled from $\mathsf{Lap}(\Delta K/\epsilon)$ and proofed that this connection between the noise and the sensitivity of $K$ satisfies $\epsilon$-differential privacy. The actually perturbed output $\hat{K}(D)$ is generated by summing up the output of $K$ and the noise. $\hat{K}(D) = K(D) + \mathsf{Lap}(\Delta K/\epsilon)$.

Unfortunately, in our scenario the sensitivity of the function $K$ is the maximum distance between any two possible genomes, *i.e.*, the length of the genomes themselves in case of the edit distance metric. Following this definition, the probability of any query result (edit distance) — regardless of the genomes — may change

by at most a factor of $\exp(\epsilon)$. Smaller values for $\epsilon$ increase privacy, but also potentially decrease utility. How to chose a trade-off is discussed by Lee and Clifton [LC11]. On the opposite, increasing values for the sensitivity $\Delta K$ let the Laplace distribution approach the uniform distribution.

As the sensitivity in our case is rather large, this clearly annihilates the utility of any differentially private query result, since the resulting noise distribution quickly approaches the uniform distribution with decreasing $\epsilon$. Furthermore, we want to protect the presence of a genome in the database against a series of edit distance queries with other genomes. In repeated queries the security parameter $\epsilon$ adds up, since the same genome in the database may be queried [McS10]. Frameworks offering basic differential privacy functionality are already available from McSherry [McS10] and Roy et al. [Roy+10], but suffer from side-channel leakage, which was only partly fixed by Haeberlen, Pierce, and Narayan [HPN11].

Based on the inherent limitations of differential privacy in our setting, we therefore propose in this chapter the competing approach of detecting possible inference attacks based on the similarity of queries.

Further attacks on privacy mechanisms in genomic computing have to be considered. Bloom filter matching using the approach of [BC04] (keyed cryptographic hash functions instead of regular hash functions) is insecure [Kuz+11]. A sophisticated attack can infer information from a Bloom filter with unknown hash functions using environmental information, such as the space of all genomes. We therefore use homomorphic encryption to protect the Bloom filter and are not susceptible to this attack. Anonymization techniques have also been found to be insecure [Wan+09b].

A related problem to privacy-preserving genome matching between two parties is outsourcing of this computation. This has been first considered in [AL05]. A protocol for two servers executing a secure computation is presented. The protocol of Troncoso-Pastoriza, Katzenbeisser, and Celik [TKC07] has been used for outsourcing in [BA10]. The protocol of Jha, Kruger, and Shmatikov [JKS08] has been used in [Bla+12a]. A clever technique of partitioning the problem into a coarse and a fine-granular part has been presented in [Che+12] to solve the problem of read mapping. That is matching a private genome against a public reference genome. An approach for simple queries on an encrypted, outsourced genome database are presented in [Kan+08].

## 5.3. Preliminaries

We quickly recall necessary building blocks previously introduced in Chapter 3, as well as introduce new ones and describe necessary modifications of well known primitives.

### 5.3.1. Homomorphic Encryption

The definition in Section 3.5 is used throughout this chapter and recalled briefly using a simplified notation. Specific properties — necessary for the construction and security proofs — are also introduced in this section. Homomorphic encryption supports a homomorphism of (at least) one arithmetic operation on the ciphertexts to an arithmetic operation on the plaintexts. Additive Homomorphic Encryption (AHE) supports addition as the homomorphic operation on the plaintexts. Let $E(x, r)$ denote the encryption of plaintext $x$ with randomization parameter $r$. Then the following addition properties hold

$$E(x, r)E(y, s) = E(x + y, rs) \tag{5.1}$$
$$E(x, r)^y = E(xy, r^y) \tag{5.2}$$

For readability reasons we write the simplification of $E(x) = E(x, r)$, which also always uses a fresh $r$ implicitly. Particularly we use the encryption scheme of Boneh, Goh, and Nissim [BGN05]. This scheme is somewhat homomorphic, because it also supports one round of multiplication. Let $E'(x, r)$ denote a second encryption of plaintext $x$ with randomization parameter $r$. This scheme is constructed from a bilinear map $\hat{e}(c, d)$. Let $\eta$ be a fixed parameter and $\theta$ a randomization parameter[1], then the following multiplication property holds

$$\hat{e}(E(x, r), E(y, s)) = E'(xy, \eta + xs + yr + \theta rs)$$

For the encryption $E'(\cdot, \cdot)$ the above addition properties hold again.

The Boneh-Goh-Nissim (BGN) scheme has been proven to be Indistinguishability under Chosen Plaintext Attack (IND-CPA) secure under the subgroup decision problem by Boneh, Goh, and Nissim [BGN05]. This indistinguishability implies that an adversary cannot distinguish a ciphertext even if it is from the same plaintext without the private key. Somewhat Homomorphic Encryption (SHE) is less powerful than Fully Homomorphic Encryption (FHE) which supports arbitrary operations on finite fields, but also significantly more efficient, since it does not require the error-correction operation.

The BGN scheme uses asymmetric keys and in our scenario the server has only the public key, while the client holds the private key. This means the client can perform encryption $E(x)$ and decryption $D(c)$ whereas the server can only perform encryption $E(x)$. This implies that all ciphertexts remain indistinguishable to the server unless the plaintext is explicitly disclosed in the protocol.

[1] If we choose the generators in the group of the result of the bilinear map cleverly, then we can compute $\eta$ and $\theta$.

### 5.3.2. Fuzzy Commitment

A commitment scheme binds the committer to a certain value (binding property) without revealing its value (security property). A fuzzy commitment scheme binds the committer to a value within a distance of the committed value. The distance metric can again be the Hamming distance. The fuzzy commitment scheme of [JW99] commits to a randomly selected codeword $y$ and secret key $\kappa$ with $\mathsf{MAC}(y, \kappa)$ and adds a distance sequence $\tau$. Let $v = y \oplus \tau$ be the committed value. When testing whether a value $z$ is covered by the commitment, one first computes $z' = z \oplus \tau$ and then performs error-correction to $z''$ of $z'$. If $\mathsf{MAC}(z'', \kappa) = \mathsf{MAC}(y, \kappa)$, then $z$ is covered by the fuzzy commitment of $v$.

This type of fuzzy commitment is perfectly suitable for privacy-preserving storage of biometrics, but in our application we require the element in the commitment to be checkable (and not random). More similar to [DFM98], we treat the value to be committed as the $l$-bit string $y'$. First, $y'$ is decoded to the information word $x$ of length $l'$ and $x$ is committed using a MAC function ($\mathsf{MAC}(x, \kappa)$). To check whether a $l$-bit string $y''$ is covered by the commitment to $x$, one performs decoding to the information word $x''$ and then checks whether $\mathsf{MAC}(x, \kappa) = \mathsf{MAC}(x'', \kappa)$.

In our construction, Alice commits to her genome sequence using this commitment scheme with a pre-defined MAC function (see Section 5.5.1), *i.e.*, she sends $\mathsf{MAC}(x, \kappa)$ along with the encrypted Bloom filter. If Alice submits two Bloom filters for genome strings within a certain distance, she likely commits to the same value $x$ using the fuzzy commitment scheme. Bob will be able to detect this and can withhold his answer.

Note that in our construction of the fuzzy commitment using modified, first-order Reed-Muller codes, it is possible that two committed values $y'$ and $y''$ are close, but are decoded into different information words and consequently fuzzy commitments. This is unavoidable except in a perfect code which only exists for $t = 1$ or $t = 3$, but results in a false-negative. Similarly, it is possible that two committed values are not close, but are covered by the same fuzzy commitment generating a false-positive. We empirically estimate these errors in our experiments and discuss techniques to reduce false-positives and -negatives.

## 5.4. Design

The basic protocol for matching sequences is taken from Chapter 4 and enhanced by necessary elements to achieve the claims made in Section 5.1. To achieve this,

we first recall the basics of the comparison scheme from Chapter 4 and introduce changes in the proposed protocol to allow detection of a Honest-But-Curious (HBC) client (Alice) that performs an inference attack. To strengthen the detection algorithm, Zero Knowledge Proofs (ZKPs) are presented to allow detection of malicious adversaries performing an inference attack.

An inference attack detection scheme that compares the similarity of user inputs can perform detection either on a per-user basis, across a group of users or all users. Furthermore, if detection is performed only for queries from the same user, how to handle Sybil attacks on the system? Our scheme resides in the centralized setting with global knowledge, *i.e.* the party supposed to defeat Sybil attacks is a central server who knows all users, which are always directly connected to him in a star topology. Defeating Sybil attacks in such a scenario could be implemented by simply requiring all users to register personally for an account and link accounts to identity documents. There are also other, non-trivial methods which try to detect Sybils [Vis+10], but require assumptions about the behavior of the attacker or knowledge about the social graph. We therefore don't consider Sybil attacks. Furthermore, collusion attacks are also out of scope similarly to Sybil attacks. In our scheme attacks are evaluated on a per-user basis, not across users.

### 5.4.1. Genome Matching Using Bloom Filters

We approach the problem of comparing two genomes by first converting each into a Bloom filter. Bloom filters, as described in Section 3.4, can be used to represent sets and run member tests against them. When using Bloom filters with equal configuration, *i.e.* identical values for $l, k$ and functions $h_1(), \ldots, h_k()$, set union and intersection can be performed through the use of bit-wise AND and OR operations. Building upon this a bit-wise XOR ($\oplus$) can be used to calculate the Hamming distance between set representations and thus approximate the symmetric difference between sets. The cardinality of the symmetric distance is an appropriate measure to solve the original string similarity problem as it closely approximates the edit distance — as shown in Chapter 4. For brevity, the protocol is only sketched and further information can be found in the mentioned chapter.

A similar analysis, which also demonstrates the utility of using Bloom filters for string comparisons was done by Durham et al. [Dur+12]. However, they only discuss privacy as mutual information which is leaked. Different techniques to estimate the string similarity are evaluated regarding their performance, accuracy and information leakage. No further privacy protection was applied. Figure 5.1 recalls the beginning of the string matching protocol from Chapter 4. It is enhanced with hooks in places that are changed by the scheme described in this chapter. Specifically these places are denoted "Build Fuzzy Commitment" and

"Check Commitment", as this is the overall functionality these steps must provide. The function $\mathsf{Dec}_{\mathrm{ecc}}(b_A)$ represents the error-correction and decoding function for the underlying Error-Correcting Code (ECC).

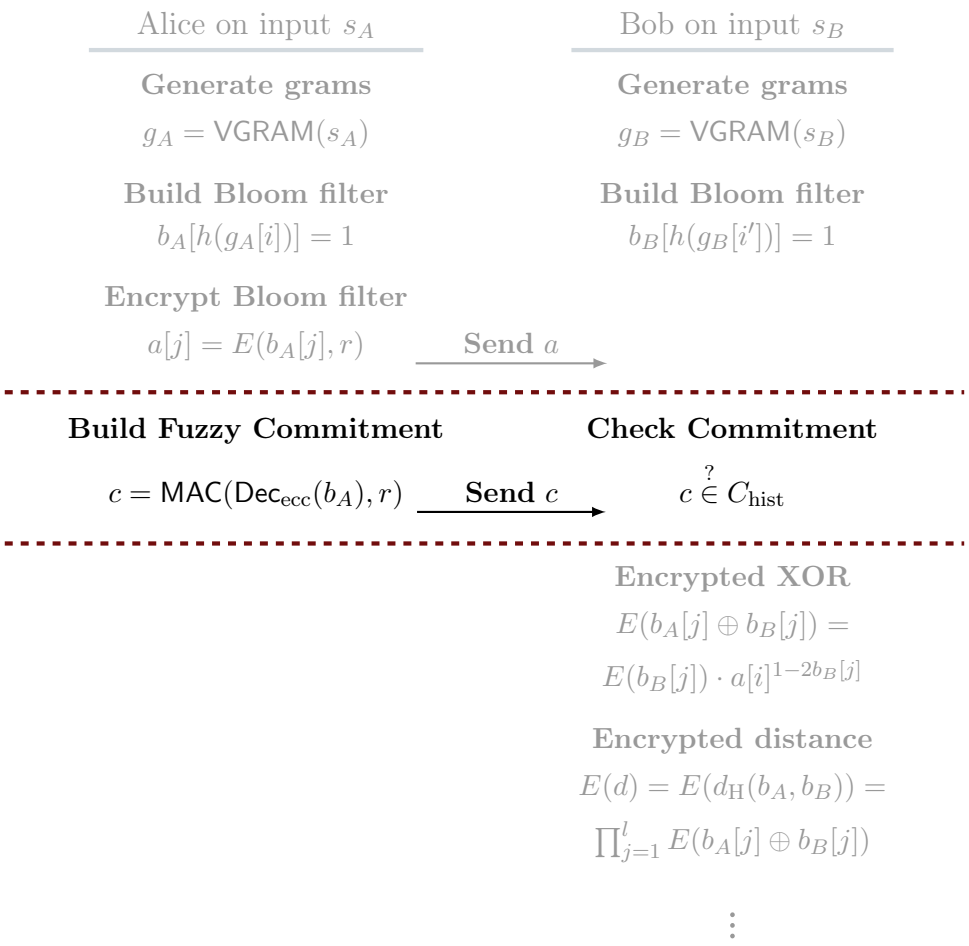



Figure 5.1.: Extension for privacy-preserving distance measure protocol presented in Chapter 4. The client generates a fuzzy commitment of its input (see Section 5.3.2), which is checked at server side against the history of all previous query commitments $C_{\mathrm{hist}}$. Similar to the concept of Locality-Sensitive Hashes, presented in Section 2.1.1, close inputs are mapped to the same commitment.

**Changes from Chapter 4** The depicted protocol follows the approach from Section 4.2. The Bloom filter configuration was modified towards using a false-positive probability of $p_{\mathrm{fp}} = 0.5$ instead of 0.1, as it was discussed in the original presentation. As a result the generated Bloom filters $b_i$ with length $l_i$ have a Hamming weight of roughly $w_{\mathrm{H}}(b_i) = {}^{l_i}/_2$ and thus a normalized Hamming weight

$\bar{w}_{\mathrm{H}}(b_i) = {}^{w_{\mathrm{H}}(b_i)}/_{l_i} \approx 0.5$. Section A.3 discusses the implications by having different false-positive rates for the Bloom filter. Particularly for 10000 samples taken from Ingman and Gyllensten [IG06], the mean ${}^{w_{\mathrm{H}}(b_i)}/_{l_i}$ amounts to 0.484 with a standard deviation of 0.0059. The Hamming weight of the Bloom filters $w_{\mathrm{H}}(b_i)$ is important for the employed ECC, as it works best when bit strings with a roughly equal amount of zeroes and ones are used for error-correction towards a codeword. Almost all codewords from a first-order Reed-Muller code with configuration $m$ — and thus a codeword length of $2^m$ — have a weight of $2^{m-1}$, except for the codewords generated from the information word $(0, \ldots, 0)$ with codeword weight 0 and $(1, 0, \ldots, 0)$ with codeword weight $2^m$.

Another positive effect is the shortened Bloom filter length $l$ compared to the original work in [BK13]. For full mitochondrial DNA sequences with approximately 16568 nucleotides, a Bloom filter length of 23905 bit is used instead of the originally proposed 157261 bit, as can be derived from Equation (3.2). In return the number of homomorphic operations on the client and server side is greatly reduced. Especially the time used for encrypting the Bloom filter at the client makes up for the largest part of the total protocol time. The mean time for comparing a full sized mitochondrial genome is therefore lowered by 77% compared to [BK13] for an otherwise identical configuration.

### 5.4.2. Inference Attack Detection

As indicated by Figure 5.1, the protocol from Chapter 4 is extended by the creation of a fuzzy commitment for the query on client side and the check if this commitment was previously seen on the server side. Section 5.3.2 describes a fuzzy commitment and its use case. Within this protocol the fuzzy commitment groups similar inputs — similar Bloom filters — and maps them to the same commitment, comparable to the mapping Locality-Sensitive Hashing (LSH) performs. Similarity is defined by Bloom filters, which decode to the same information word by the employed error-correction scheme. Thus the actual additional steps at the client side are:

- Decoding of Bloom filter $b_A$ into information word $x$
- Generate commitment $c = \mathsf{MAC}(x, r)$ for $x$ using an appropriate commitment scheme (see Section 5.3.2 and 5.5.1)

The first two steps are performed together within the decoding function of our selected ECC $x = \mathsf{Dec}_{\mathrm{ecc}}(b_A)$, while the last step generates our commitment $c = \mathsf{MAC}(\mathsf{Dec}_{\mathrm{ecc}}(b_A), r)$, as depicted in Figure 5.1.

Upon generating the commitment, the client sends the Bloom filter $b_A$ together with the commitment $c$ to the server. The server needs to check whether it has

already received a query for the same commitment before. It cannot test this upon the query directly, as the query is encrypted using an indeterministic, semantically secure, encryption scheme, upon which the server must not be able to detect two ciphertexts for identical plaintexts with non-negligible probability in the size of the security parameter. The commitment however must be deterministic to be able to match two identical commitments. Section 5.6 will discuss security and privacy implications that follow from the deterministic commitment.

If the commitment was not seen before, it is added to the commitment history $C_{\text{hist}}$ of Alice to be checked against future queries. However, in case the commitment has been used previously, the query is dropped and the protocol execution is aborted. Due to this behavior Alice cannot easily use close queries to infer information, as close queries will be detected and dropped.

The described protocol works as long as the client follows the HBC model, which dictates the correct execution of protocols and algorithms, using correct inputs. However, once a client wants to perform an inference attack, it might also be willing to deviate from the protocol and adjust inputs arbitrarily. The problem that arises in such a setting of course is, whether the client has committed to the same value that is used to query the server. To let the client prove usage of identical inputs, we present techniques for detecting deviations from the protocol and input done by the client. That is, over the next sections we describe changes to the steps described above to make inference detection support malicious client adversaries.

## 5.5. Zero Knowledge Proof

In this Section we describe the zero-knowledge proof (ZKP) that the client Alice submitted the same Bloom filter in the homomorphic encryption and (corresponding) information word in the fuzzy commitment scheme. Obviously, the generic technique for ZKPs based on Karp reductions to languages in NP is incredibly inefficient. We therefore provide a specialized proof for our problem based mostly on somewhat homomorphic encryption. Our choices of ECC in Section 3.7 and MAC function in Section 5.5.1 have been specifically tailored to our choice of homomorphic encryption scheme, *i.e.*, they are computable in the efficient scheme of Boneh-Goh-Nissim (see Section 5.3).

Still, it is quite difficult to prove that an information word has been correctly decoded from a Bloom filter using homomorphic encryption. In Reed-Muller codes this requires $l'$ majority (range) proofs. Instead, we first compute the codeword $d$ for the Bloom filter, prove that it has the right distance from the Bloom filter in Section 5.5.2 and decodes to the information word $x$ in Section 5.5.3. Specifically,

to compute the codeword $d$, we decode the Bloom filter into $x$ and apply the encoding function of the chosen ECC to generate the codeword $d = \mathsf{Enc}_{\mathrm{ecc}}(x)$ This requires only one range proof and several proofs of plaintexts zero and is therefore significantly more efficient. Finally, in Section 5.5.4 we prove that the information word has been used in the fuzzy commitment. Note that neither the codeword nor the information word is seen in the clear by the server, but its computation can be performed in the clear by the client without homomorphic encryption.

In our construction we will use ZKPs that a ciphertext is an encryption of either of two plaintexts from [DJ01] and that a vector of ciphertexts is a shuffle of another vector of ciphertexts [Gro10]. Furthermore, we use a simple technique to prove that a plaintext $x$ is encrypted by ciphertext $E(x, r)$ as described in Section 3.5: The prover reveals the random parameter $r$.

Before we start describing the actual parts of the overall proof, we first fix the MAC function that will be used in the fuzzy commitment scheme. The choice of MAC function is influenced by the requirements of the overall ZKP as detailed later. The subsequent sections show how to prove input equality between encrypted Bloom filter and MAC function. Finally the single proofs are combined.

### 5.5.1. Message Authentication Code Function

We replace the use of a standard MAC function in fuzzy commitments by a one-way function, as the strong security properties of forgery resistance or pseudo-randomness are not required. The one-wayness property is sufficient, since we sample from a sufficiently large, but restricted domain of $l'$-bit information words[2]. Usually symmetric cryptography is used for speed, but the time-critical operation in our case is the ZKP. Therefore we use a one-way function whose properties can be proven.

We use the homomorphic encryption function with a fixed random parameter $\kappa$. Using a domain with more than $l'$ bits in the homomorphic encryption function we are certain to avoid collisions. If the information word has more bits than the domain, we can use multiple encryption functions. Then our one-way ("MAC") function is

$$\mathsf{OWF}(x, \kappa) = E(x, \kappa)$$

The key $\kappa$ is at Alice's discretion, but she has to commit to it by sending a random value $s$ and $\hat{s} = E(s, \kappa)$ to Bob during setup. Of course, Alice has to also

[2] Alice only needs to decrypt values smaller than the logarithm of the size of the domain.

prove proper construction of the homomorphic encryption scheme. Since the key is a composite of two primes, she can use another zero-knowledge proof, *e.g.* [CM99], or for efficiency reveal the primes to an authorized third party. Alice also needs to prove that the generators of the encryption scheme were chosen randomly. She can do this in a white-box way by revealing the randomness used to compute them. Note that all of these steps only need to be completed once during setup and not for each genome Alice is querying.

### 5.5.2. Proof of Hamming Distance

Alice intends to prove that the same Bloom filter was used in the homomorphic encryption and in the fuzzy commitment scheme, but the information word in the fuzzy commitment scheme is based on an error-corrected Bloom filter. This error-corrected Bloom filter is a codeword. Therefore there will likely be a difference between the Bloom filter and the codeword, but the difference must be small. Alice therefore proves that the Hamming distance between the two strings is at most our chosen parameter $\delta$.

Let $b_A[i]$ $(0 \leq i < l)$ be the $i$-th bit of the Bloom filter $b_A$ in the homomorphic encryption. Let $r[i]$ be fresh random values and $a[i] = E(b_A[i], r[i])$. Alice sends all $a[i]$ to Bob.

Alice first creates the codeword $d$ by error-correcting the Bloom filter $b_A$. As described before, using the Reed-Muller ECC to generate a codeword $d$ for a give bit string $b_A$ implies decoding it to an information word $x$ followed by an encoding to $\mathsf{Enc}_{\mathrm{ecc}}(\mathsf{Dec}_{\mathrm{ecc}}(b_A))$. $d$ also has a length of $l$ bits, with $d[i]$ again denoting the $i$-th bit of $d$. Then $d_{\mathrm{H}}(b_A, d) \leq \delta$. Let $r'[i]$ be fresh random values and $c[i] = E(d[i], r'[i])$. Alice sends all $c[i]$ to Bob.

Alice proves in zero-knowledge that all $a[i]$ and $c[i]$ are indeed encrypted bits, *i.e.*, their plaintext is either 0 or 1. She uses the ZKP from [DJ01].

Alice and Bob compute the encrypted *exclusive or* via element-wise subtraction and squaring of the encrypted bit vectors $a$ and $c$. The Hamming distance then follows from summing up all *xor*ed elements.

$$E'(d_{\mathrm{H}}(b_A, d), r) = \prod_{i=0}^{l} \hat{e}(a[i]c[i]^{-1}, a[i]c[i]^{-1})$$

They compute for each $0 \leq j \leq \delta$:

$$f[j] = E'(g[j], r') = E'(d_{\mathrm{H}}(b_A, d), r)E'(-j, 0)$$

Note that for $j = d_{\mathrm{H}}(b_A, d)$ it holds that $g[j] = 0$ and $g[j] \neq 0$ otherwise. That is, $g$ is a vector of length $\delta + 1$ and contains the elements

$$g = (d_{\mathrm{H}}(b_A, d), d_{\mathrm{H}}(b_A, d) - 1, d_{\mathrm{H}}(b_A, d) - 2, \ldots, d_{\mathrm{H}}(b_A, d) - \delta),$$

while $f$ is a vector of the same length as $g$, but all elements encrypted.

Alice performs a random shuffle of all $f[j]$. Let $f[\pi(j)]$ denote the elements in this shuffle. Alice sends the shuffled $f[\pi(j)]$ and proves in zero-knowledge that it indeed is a shuffle. Alice then reveals $f[\pi(d_{\mathrm{H}}(b_A, d))] = E'(0, r'')$ and $r''$ to Bob. Bob verifies that $f[\pi(d_{\mathrm{H}}(b_A, d))]$ is in the shuffle and an encryption of zero (with the revealed random parameter $r''$).

Bob knows that $d$ is a bit string within Hamming distance $\delta$ of Bloom filter $b_A$.

### 5.5.3. Proof of Code and Information Word

In order to prove that $d$ is a codeword with information word $x$ Alice needs to prove that all decoding equations have the same result for each bit. This follows from the construction of Reed-Muller codes [Mul54; Ree54]. The generator matrix and therefore the equations are known to both — Alice and Bob. We operate (mod 2), *i.e.*, all operations in the equations are *xor*.

Alice and Bob build all equations for each information word bit $x[i]$. Let $x'[j]$ $(0 \leq j < 2^{m-1})$ be the result of the $j$-th equation for $x[i]$. Alice and Bob can compute the result $E(x'[j])$ from $E(d)$ using the homomorphic operation. Recall that we use modified Reed-Muller codes and further utilize only first-order equations, *i.e.* two decoding bits. Let $d[j']$ and $d[j'']$ be the two bits decoding to $x'[j] = d[j'] \oplus d[j'']$. Then

$$\begin{aligned} E'(x'[j], r'[j]) = \hat{e}(&E(d[j'], r')E(d[j''], r''), E(1,0)) \\ &\hat{e}(E(d[j'], r'), E(d[j''], r''))^{-1}. \end{aligned}$$

Alice also sends the bit-wise encrypted information word $E(x[i], r[i])$ $(0 \leq i < l')$ to Bob. Alice and Bob compute the second ciphertext for each bit $E'(x[i], r'''[i]) = \hat{e}(E(x[i], r[i]), E(1,0))$. Then, Alice and Bob compute the differences between each bit of the information word and each decoding result (in one variable):

$$\begin{aligned} E'(x''[i], r''[i]) &= \prod_{j=1}^{2^{m-1}-1} (E'(x[i], r'''[i])^{-1} E'(x'[j], r'[j]))^{2^j} \\ &= E'(\sum_{j=1}^{2^{m-1}-1} 2^j (x'[j] - x[i]), r''[i]). \end{aligned}$$

This difference must always be zero and Alice proves that all $x''[i] = 0$ by revealing all $r''[i]$. Bob knows that $x$ is the information word for codeword $d$.

### 5.5.4. Proof of One-Way Function Computation

Alice sends $\mathsf{OWF}(x, \kappa) = E(x, \kappa)$ to Bob, which is referred to as $E(x', \kappa)$. Alice has already proven that $x$ is the information word for a codeword $d$ within Hamming distance $\delta$ of $b_A$. She needs to prove that the OWF was computed on $x$. Particularly, note that Alice could cheat on using the correct key $\kappa$.

Alice and Bob compute

$$E(x, r) = \prod_{i=0}^{l'-1} E(x[i], r[i])^{2^i}$$

and

$$E\big(h, r'\big) = E(x, r)E\big(x', \kappa\big)^{-1}.$$

Alice proves that $h = 0$ by sending $r'$. Bob verifies that indeed $E(h, r') = E(0, r')$.

Recall that $\hat{s} = E(s, \kappa)$ is Alice's commitment to her OWF key $\kappa$. Alice needs to prove that $E(x, \kappa)$ was also computed with the random parameter $\kappa$. Alice and Bob compute

$$E(u, v) = \mathsf{OWF}(x, \kappa)E(s, \kappa)^{-1}.$$

If the OWF was computed properly, then $v = 0$ (but $u$ is randomly distributed). Alice needs to prove $v = 0$, but without revealing the difference of $d$ and $s$. She therefore cannot reveal $u$, as $u$ is the difference between $x$ and $s$, with $x$ being the decoded information word obtained from the codeword $d$. Revealing $u$ would thus also reveal the difference between $d$ and $s$.

Instead, she chooses a random value $r''$ and computes $w = E(r'', 0)$. She sends $w$ to Bob. Bob flips a coin $\eta \in \{0, 1\}$ and either challenges Alice to reveal $r''$ (if $\eta = 0$) or $r'' + u$ (if $\eta = 1$). Bob verifies that $w = E(r'', 0)$ (in case of $\eta = 0$) or $wE(u, v) = E(r'' + u, 0)$. This is repeated $\lambda$ times. Alice's chance of successfully cheating is $2^{-\lambda}$.

## 5.6. Security Analysis

**Theorem 5.1**

Our ZKP is complete, sound and honest-verifier zero-knowledge.

Our ZKP is a composition of several smaller ZKPs. We therefore either prove their completeness, soundness and zero-knowledge properties first or reference the relevant literature.

We begin in reverse order with the ZKP whether $v = 0$ in $c = E(u, v)$. It is *complete*, because if $v = 0$, then in both options — $E(r'', 0)$ and $E(r'' + u, 0)$ — the randomization parameter is 0. It is *sound*, because if $v \neq 0$, then either in $E(r'', v')$ or $E(r'' + u, v'')$ the randomization parameter — $v'$ or $v''$ — is not 0. The prover will be caught with probability $\frac{1}{2}$. It is *honest-verifier zero-knowledge*, because the following simulator does not require the secret input, *i.e.*, the value $u$ or the secret key. First, the simulator outputs an uniformly chosen $\rho$ as the last message. If the verifier selects $\eta = 0$, it sends $E(\rho, 0)$ as the first message. If the verifier selects $\eta = 1$, it sends $w = E(\rho, 0)E(u, v)^{-1}$ as the first message.

Next, we consider the ZKP whether $x = 0$ in $c = E(x, v)$. The completeness, soundness and zero-knowledge property of this ZKP were given in [DJ01]. Based on this, the proofs for one of two plaintexts were also given in [DJ01]. Furthermore, the completeness, soundness and zero-knowledge property of the shuffle ZKP were given in [Gro10].

We need to prove the completeness, soundness and zero-knowledge property of the composite ZKP. For brevity we do not prove each step individually, but only present the summary of our construction. We follow the composition theorem for semi-honest secure computations by Goldreich [Gol04] and replace the sub-ZKP by oracle functionality. The composite ZKP is *complete*, because if the fuzzy commitment $\mathsf{OWF}(x, \kappa)$ and the encrypted Bloom filter $a[j]$ $(0 \leq j < l)$ relate to the same information word $x$, all oracle ZKPs will succeed. The composite ZKP is *sound*, because if the fuzzy commitment $\mathsf{OWF}(x, \kappa)$ and the encrypted Bloom filter $a[j]$ use non-related information words, one oracle ZKPs will not succeed. This is an abbreviation, since we would need to show how the prover could deviate and then be caught. Instead — for brevity — we leave this as a proposition.

In order to show honest zero-knowledge we give the following simulator. First, Alice sends the encrypted Bloom filter $a[j]$ and $c[j]$ $(0 \leq j < l)$. These can be simulated using random plaintexts and randomization parameters, since the encryption is IND-CPA secure [BGN05]. We invoke the simulator for ZKP that the plaintext is one of either two from [DJ01]. Note that this simulator will be correct, since it is independent of the secret input. The subsequent computations are performed by both — Alice and Bob. We then choose $\delta + 1$ random ciphertexts including $E'(0, r'')$. We invoke the simulator for the shuffle from [Gro10]. Again, recall that the simulator is independent of the secret input. Furthermore, due to the IND-CPA security our simulated shuffle is indistinguishable from a real shuffle. The simulator reveals the random parameter $r''$.

The simulator computes the information word $x$ for the (random) codeword $d$ ($c[j] = E(d[j], r'[j])$ for $0 \leq j < l$). It chooses random $r[i]$ ($0 \leq i < l'$) and outputs the ciphertexts $E(x[i], r[i])$ for the information word. Again, these are IND-CPA indistinguishable. Furthermore, it performs the computations as in Section 5.5.3 and outputs the corresponding $r''[i]$.

The simulator chooses random $r'$ and sends from the previous simulated messages $E(x, \kappa) = \prod_{i=0}^{l'-1} E(x[i], r'[i])^{2^i} E(0, r')^{-1}$ as the fuzzy commitment. It is indistinguishable again due to the IND-CPA security of the encryption scheme. The subsequent computations are again performed by both — Alice and Bob. The simulator reveals $r$. The simulator invokes the simulator for the ZKP whether $v = 0$ in $c = E(u, v)$ from above. Again, this simulator succeeds, since it is independent of the secret input.

We have given a complete simulator — independent of the secret input by Alice — for the message exchange between prover and verifier. It uses the simulators of the oracle ZKPs as subcomponents.

**Information Leakage Towards Server** Since all fuzzy commitments from Alice use the same randomization parameter $\kappa$ — which Alice commits to during setup — the commitments are deterministic and can be compared on server side. This property is essential for efficient history checking as described in Section 5.4.2. However, deterministic queries are also apt to leak frequency information and possibly allow dictionary attacks. Both concerns however do not pose a privacy issue in our system for two reasons. First, the randomization parameter of $\mathsf{MAC}(\cdot, \cdot)$ is at the client's discretion, thus the server cannot generate and check commitments. Second, for frequency attacks to be mounted, the attacker needs frequency information about queries. Without such a distribution — *i.e.* if every commitment occurred only once — the server cannot infer information from the commitments of queried strings. Achieving unique commitments for genuine queries and identical commitments for close inference attack queries with a low false-positive and -negative rate is one of the goals of the proposed scheme. Reaching this goal depends on the configuration of the ECC and its usage, as will be discussed in Section 5.7.3.

## 5.7. Theoretical Evaluation

Goodrich [Goo09] describes two attacks to discover private strings using similarity scores obtained in a supposedly privacy-preserving way. One of the two attacks Goodrich describes relates to the black-peg score of the Mastermind game and can

easily be adopted to be used against edit distance scores. The second attack uses results from a privacy-preserving sequence alignment, which is not appplicable in our case. We therefore focus on the straight-match (black-peg) score attack. Again, for brevity we only give an overview of the attack and refer the reader to [Goo09] for details.

We are going to describe an example of a class of efficient inference attacks formulated by Goodrich [Goo09] in Section 5.7.1. A generalization of the proposed attack, which can handle rejected comparisons is presented in Section 5.7.2. In Section 5.7.3 we describe a property of the Mastermind attack, which is responsible for a certain pattern observed for queries that are part of an Mastermind inference attack. We describe how to exploit that attack structure to detect Mastermind and related attacks. Section 5.7.4 analytically evaluates sensible configurations for the underlying Reed-Muller ECC with respect to trade-off detection accuracy of inference attack queries against the false classification of genuine queries as an ongoing attack.

### 5.7.1. Description of the Original Attack

The Mastermind attack presented by Goodrich [Goo09] aims to efficiently infer a private, real-world sized genomic sequences $s$ from another party. The only prerequisite is that a privacy-preserving protocol $\pi$ can be used to compare an arbitrary sequence $r$ over the same alphabet against the private sequence. The protocol returns the actual distance $d = \pi(r, s)$ or an approximation of it. A possible distance measure could be the edit distance.

The generic algorithm to infer the private sequence $s$ starts with a reference string $r_0$ and an alphabet $A$, containing $j = |A|$ characters. These are used to generate $j - 1$ strings $\{r_1, \ldots, r_{j-1}\}$ in such a way that two characters taken from the same position of different strings are always pair-wise different. The strings $r_i$ $(1 \leq i < j)$ are deviations from $r_0$. Furthermore, $r_0$ defines the base character for a specific position. Let $r_i[p]$ denote the character at position $p$ in string $r_i$. For each string $r_i$ $(1 \leq i < j)$, the character at position $p$ is selected from $A \setminus \{r_0[p], r_1[p], \ldots, r_{i-1}[p], r_{i+1}[p], \ldots, r_j[p]\}$. It follows that for any position $p$, $\bigcup_{i=0}^{j-1} r_i[p] = A$. The method for selecting the actual Character has no impact on the effectiveness of the attack. It could be chosen uniformly at random, sequentially ordered by character code or any other mean. However, it is important that all $r_i[p]$ are distinct and thus present a partition of $A$.

For each string $r_i$ $(0 \leq i < j)$ a similarity score $d_i = \pi(r_i, s)$ using the privacy-preserving comparison protocol $\pi$ is obtained and the algorithm starts a recursion for each $r_i$ in which it performs the following steps. These steps are also visualized in Figure 5.2.

1. Split the previously queried strings into a left and right part $r_i = r_{iL}|r_{iR}$, with "|" denoting the string concatenation.

2. Select the left part (*e.g.* $r_{iL}$) and replace it by another string $r'_{iL} \in A^{\|r_{iL}\|}$. Let $\|r\|$ denote the length of the string $r$. The replacement is performed $j-1$ times, generating $j-1$ new strings (called $r_4, r_5, r_6$ in Figure 5.2). All $(r'_{iL}[p])$ at position $p$ across all substrings generated in this step (including the original substring $r_{iL}$) must be distinct. All distances (*i.e.* $d_4, d_5, d_6$) for the new constructed strings are queried using $\pi$.

3. Distances of the changed parts $d_{i\mathrm{L}}$ and $d_{i\mathrm{R}}$ — in the example $d_{4\mathrm{L}}$ and $d_{4\mathrm{R}}$ — can be calculated as $d_{i\mathrm{L}} = d_p - d_i$ and $d_{i\mathrm{R}} = d_i - d_{i\mathrm{L}}$. $d_p$ references the previously calculated distance for the parent node in the Mastermind recursion tree, as visualized in Figure 5.2.

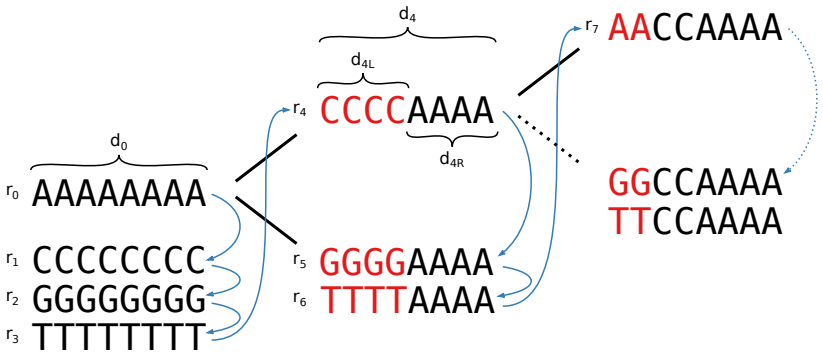



Figure 5.2.: Visualization of Mastermind attack scheme. $r_i$ denotes the sequences that are queried, while $d_i$ denotes the equivalent distances returned. $d_{i\mathrm{L}}$ ($d_{i\mathrm{R}}$) denotes the calculated distance for the left (right) — changed (unchanged) — substring. Edits made by the algorithm are depicted in red, while the blue arrows show the sequence of queries.

**Stop Condition** If at some point the similarity score equals zero for one query, then the used deviation from the original character set can be ignored for deeper recursive steps and the number of strings to be generated and queried reduces by one. At some point the number of queried strings drops below two and the recursion stops.

### 5.7.2. Goodrich Randomization

Using the proposed inference attack detection, we describe its implications on the original Goodrich attack and construct a generalization of the specific attack

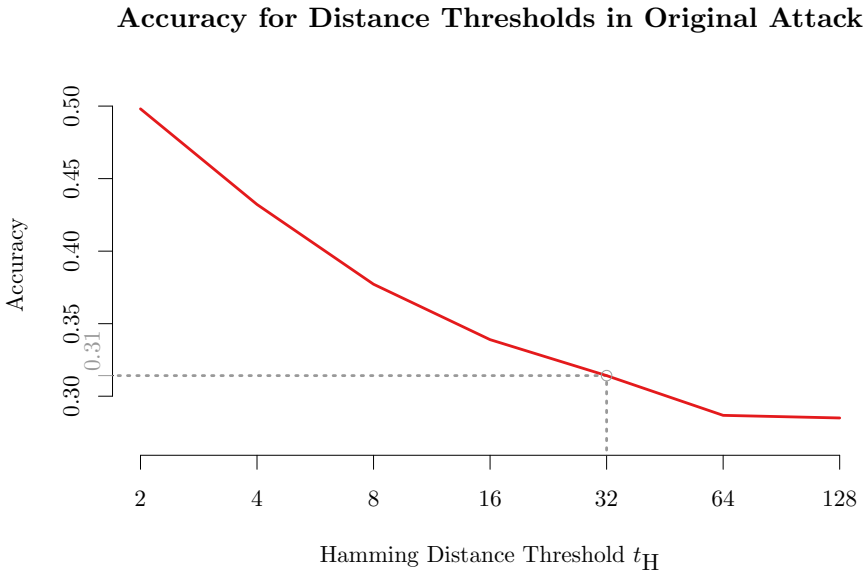



Figure 5.3.: Accuracy of original Goodrich attack using different simulated thresholds for rejecting close queries. A Hamming distance threshold $t_H$ means that any query with an Bloom filter Hamming distance less than $t_H$ to any of the previously accepted queries will be rejected. This simulation therefore does not include false-positives or -negatives in similarity detection, or other characteristics which are introduced by the ECC. Taking a threshold of $t_H = 32$ for example results in a simulated accuracy of 0.31 for the string returned by the Mastermind attack.

described by Goodrich to properly handle rejected queries, for example due to inference detection algorithms. Finally, the generalized attack is evaluated against a simplified inference attack detection construction. Analytical and empirical evaluations of the actual construction can be found in the subsequent sections.

In Section 5.3.2 we presented the use of an ECC — specifically a Reed-Muller code as given in Section 3.7 — to generate a fuzzy commitment. The benefit of having such a commitment is to detect similar queries which are covered by the same commitment.

The Hamming distance on strings of equal length equals the number of positions on which the strings differ. Depending on the configuration of the used ECC, two submitted Bloom filters within a certain Hamming distance $\delta$ can be detected as similar with high probability and no distance score will be calculated. Put

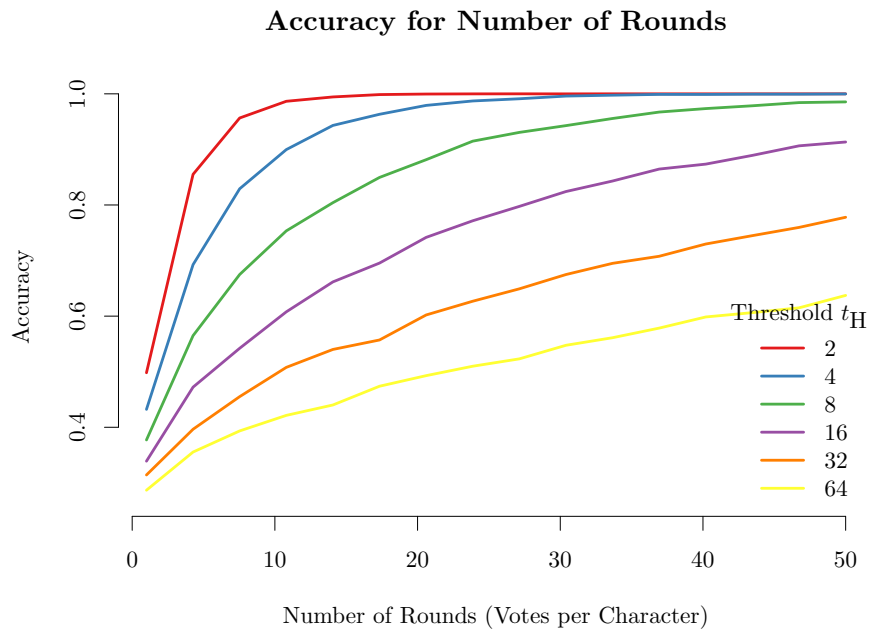



Figure 5.4.: Accuracy after several randomization rounds using the simulation setup without the ECC.

differently, for a new query $q$ and the set of all previously accepted queries $Q$, let $\delta$ be the smallest Hamming distance to all elements in $Q$: $\delta = \min_{p \in Q} d_H(q, p)$.

The Hamming distance threshold $t_H$ in the following figures describes different configurations of the inference detection algorithm. Specifically, $t_H$ describes the minimum Hamming distance between any two valid queries. Intuitively, if the Hamming distance $\delta$ for a query is below the threshold $\delta < t_H$ it should be detected and rejected with high probability, whereas a $\delta \geq t_H$ should result in an accepted query with high probability. Please note that $t_H$ is not the parameter used for configuring an ECC, but solely a threshold for the pair-wise Hamming distance between any two accepted queries. The following evaluation is based on a simulation of the inference attack detection without the use of an ECC and thus without the false-positives and -negatives introduced by the ECC. It is therefore an optimal solution in terms of query matching based on the Hamming distance. If a query was generated with a Hamming distance smaller than the threshold $t_H$ to any of the previously accepted queries, it is detected as being too close to a previous submission and the answer will be withhold. Figure 5.3 shows how the accuracy of the original Mastermind attack goes down as the Hamming distance threshold $t_H$ increases. The accuracy was calculated by comparing the proposed string by the Mastermind algorithm against the correct one. The ratio of the num-

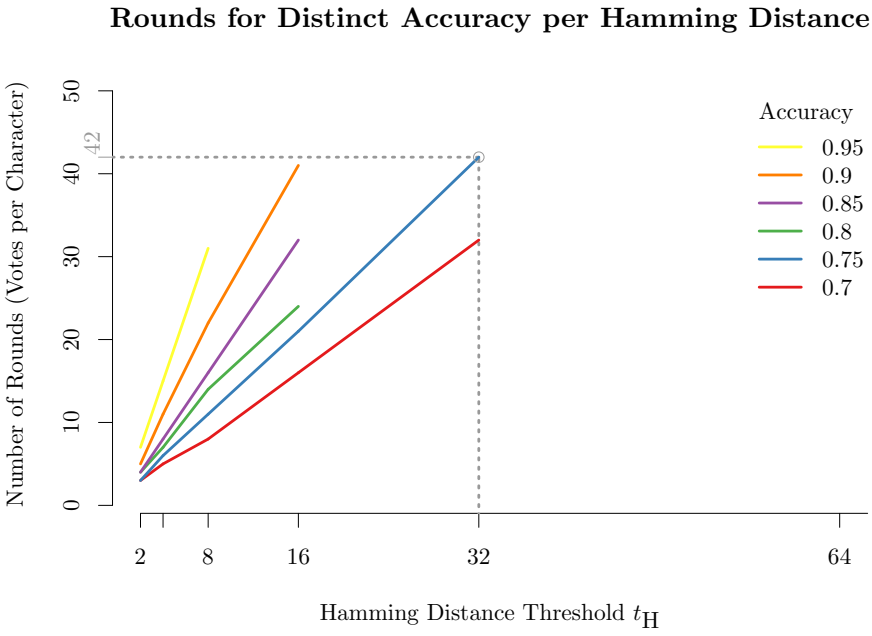



Figure 5.5.: Rounds necessary to reach distinct accuracy

ber of correctly extracted nucleotides over the total length of the sequence (approx. 16568 characters) is given as the mean accuracy value. 100 different genomes were sampled from the mtDB [IG06] with 100 runs performed for each of the genomic sequences.

Of course, an attacker is not bound to the original Goodrich attack. An adapted version, handling rejected queries, is used to test our detection mechanism. Instead of running the Goodrich attack as described (that is to abort recursion when a query gets rejected), we can calculate the probability that a certain character appears at a specific position. After all recursion paths have led to a rejection, the generalized goodrich attack starts all over using a fresh random reference string $r_0$. The substring generation and distance calculation follows the original attack. With each invocation of the adopted Goodrich attack, a vote for every possible character at every position is recorded. The adversary uses the top voted character for every position of the string to build the inferred sequence, following the plurality voting scheme. Figure 5.4 gives the accuracy (number of correctly guessed characters divided by the total number of characters) of the inferred sequence after a certain number of attack invocations (rounds). The different curves show the achieved accuracy using the adopted Goodrich attack and plurality voting under different thresholds for minimum query distances.

In contrast, Figure 5.5 specifies the number of attack rounds necessary to reach

a certain accuracy level for different detection scenarios, rejecting strings with a Hamming distance below $t_H \in \{2, 4, 8, 16, 32, 64\}$. Thus it specifies the gain achieved against attackers trying to perform inference attacks. A single round has about the same complexity and number of queries as the original Goodrich attack, which needed on average 536 queries to infer a genomic sequence. Following the results presented in Figure 5.5, an attacker that tries to infer a sequence with 95% accuracy — using the repeated, randomized attack described above — under a configuration that detects and rejects queries with a pairwise Hamming distance below 16 bits, must perform roughly $536 \cdot 60 = 32160$ queries against the database. This is 536 queries on average for a single round times 60 rounds for a 95% accuracy extrapolated from Figure 5.5.

As a countermeasure to this attack, we propose to also limit the number of total queries to a level dependent on the target accuracy. Goodrich reports that for 90% of the genomes less than 875 queries are needed. Following the simulation, if an attacker has the goal to reach a target accuracy level of 75% for an inferred sequence (given that the Hamming distance threshold is set to $t_H = 32$), it requires the attacker to perform about 42 attack rounds (or invocations) of the adopted Goodrich attack, resulting in roughly $42 \cdot 875 = 36750$ queries. The values are taken directly from Figure 5.5. This is a significant increase in allowed queries until a certain privacy level is breached — compared to the maximum number under Goodrich's attack.

Using the above described inference attack detection and query rate limiting, we achieve a much lowered success rate and at the same time higher attack costs. In subsequent sections we analyze the detection capability of our algorithm using a specific ECC.

### 5.7.3. Distinguishing Attacks from Valid Requests

We should ensure that our detection algorithm rejects as few legitimate queries as possible due to being close to each other. When comparing genomic sequences against a reference sequence usign the edit distance, it yields on average a low number of substitutions and even less insertions and deletions. Goodrich [Goo09] gives an average number of 28 substitutions for a sample set of 1000 mitochondrial genomic sequences. Of course, an algorithm that tries to detect inference attacks based on pairwise query distances and a threshold-based classification will produce false-positives and false-negatives during attack detection. A false-positive would be the classification of a genuine query as an inference attack, due to being close to a previous query. Similarly, a false-negative would be a not-detected inference attack query, which has a high pair-wise distance to all previous queries.

Over the next sections, we are going to discuss how sensible trade-offs between detection accuracy, false-positives and false-negatives can be found. The basis for the following discussion will be patterns emerging from the Goodrich attack, an analytic evaluation of the applied Reed-Muller ECC, as well as an empirical evaluation using three different ECC mapping schemes. We are going to start with the Goodrich attack patterns.

The mutated positions (for point mutations) in the genomic sequences — if compared to the reference sequence — are typically distributed over different regions, with a low probability of having consecutive changes on adjacent character positions. The generated strings from the inference attack described by Goodrich [Goo09], however, follow certain patterns, for example from dividing the sequence into substrings, as shown in Figure 5.2, changes are made at consecutive positions, rather than at distant ones. This difference in edited positions between queries has an effect on the Bloom filter generation and can still be detected from the Bloom filter Hamming distance. This reflection of the attack in the Bloom filter is used to better distinguish attacks from valid queries.

Recall the generation of positional grams, as it was described in Section 3.4 and 3.3. The VGRAM algorithm [LWY07] — sketched in Section 3.3 — also defines the generation of NAG-vectors, which describe the "number of affected grams" that follows from a certain edit operation or sequence of edit operations. Based on the number of affected grams we specified in Chapter 4 the approximated maximum Hamming distance over Bloom filters for a given edit distance.

Assume that the edit positions are at least $q_{\max}$ characters away from each other. Let $q_{\text{avg}}$ denote the mean length of a gram. One edit operation will then affect about $q_{\text{avg}}$ grams on average, *i.e.* the gram with the edit operation affecting the last character, the gram with the second to last character affected and so on. This is especially true for a fixed gram length and without skipping any grams. So each edit operation affects a set of about $q_{\text{avg}}$ grams. As long as edit positions are far away from each other (more than $q_{\max}$ characters), each edit operation adds about $q_{\text{avg}}$ new grams to the set of affected grams. For the VGRAM algorithm [LWY07] the mean number of affected grams for an edit operation is a bit lower than the mean length of a gram, as will be shown later.

For an edit distance $d_{\text{E}}$ between two strings, a rough estimation of the Hamming distance between the corresponding Bloom filters would be $d_{\text{H}} = q_{\text{avg}} \cdot d_{\text{E}}$. This is a good estimation for most genomic sequences found in the used genomic database [IG06]. The actual distances will be evaluated later as part of the empirical evaluation.

However, if $d_{\text{E}}$ changes appear consecutively, as is the case with the Mastermind attack described by Goodrich, the sets of affected grams for each edit operation

overlap to a very large extend. Thus if $d_E$ consecutive characters are changed, on average only $q_{avg} + (d_E - 1)$ grams are affected. The resulting Hamming distance between the corresponding Bloom filters represents this large overlap in affected grams. The Hamming distance for consecutive edit operations increases very slowly with additional edit operations, as the set of affected grams also grows slowly with additional consecutive edits.

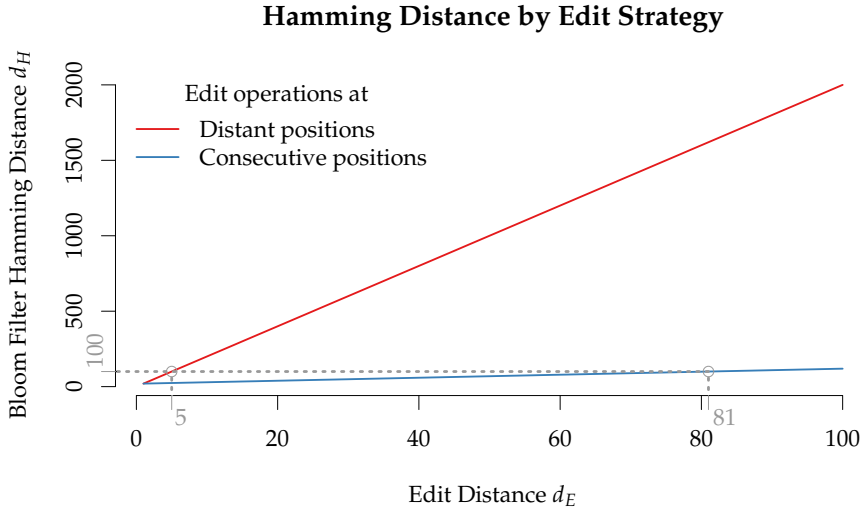



Figure 5.6.: Bloom filter changes by distance of edit positions. Edit operations at consecutive sequence positions result in a lower number of changed grams, than edit operations at distant sequence positions. A lower number of changed grams results in a lower Hamming distance between the respective Bloom filters.

Figure 5.6 visualizes both functions and it is easy to see that consecutive edit operations produce very low Bloom filter differences. The database of human genomic sequences has a large number of pairs with an edit distance of 20. Using the above approximation of $d_H = q_{avg} \cdot d_E$, such an edit distance yields an average Bloom filter distance of about 400 for a $q_{avg} \approx 20$ as taken from Figure 4.3 and Figure 4.4 in Section 4.4.1. The same edit distance applied to adjacent characters, as it is the case with the attack algorithm described by Goodrich [Goo09], however, results in a much lower Hamming distance of around 39. The curves were derived using the proposed values $q_{max} = 40$ and $p_{fp} = 0.5$, which yield a Bloom filter length of approximately $l = 23905$ bits. If a single character at a random position in the sequence is changed, the Bloom filter will have on average approximately 20 bit distance to the Bloom filter of the original string.

According to Figure 5.6 and thus the gap between changed bits for valid genomic sequences and artificial queries, we can easily define an arbitrary threshold for the

detection scheme that allows queries with higher pairwise Hamming distance. For example, if the Hamming distance threshold between submitted Bloom filters is set to $t_{\mathrm{H}} = 100$ bits, consecutive changes of up to 81 characters or changes of up to 5 characters at distant positions are detected. The just mentioned Bloom filter threshold value is an example illustrated in Figure 5.6.

This gap between valid and artificial queries can be increased even further by increasing the value $q_{\max}$ within the VGRAM algorithm and by decreasing the false-positive rate $p_{\mathrm{fp}}$ for calculating the appropriate Bloom filter length.

### 5.7.4. Configuration of the Error-Correcting Code

The previous sections presented a generic analysis and discussion of the inference detection construction without considering any specific ECC. The integrated ECC represents the core of the fuzzy mapping and detection algorithm. Therefore we must analyze what the ECC can provide, how it must be instantiated and which results can be expected in order to derive useful and practical configurations.

Our goal is to map a random bit string $b_A$ and a close string $b_B$ to the same bit string with high probability by using the error-correction and decoding function of an ECC. Closeness is defined by the Hamming distance $d_{\mathrm{H}}(b_A, b_B)$, together with some threshold $t_{\mathrm{H}} \leq t$ value. If the Hamming distance between both bit strings is above the threshold, they should decode to different bit strings with high probability. As the bit strings are practically random strings — Bloom filters generated using a cryptographic hash function —, it cannot be ensured that the ECC can correct $t_{\mathrm{H}}$ errors applied upon them. It thus follows that the decoding into identical and different information words is probabilistic. This section will provide an analysis of the properties of the selected Reed-Muller ECC. As defined in Section 3.7, a first-order Reed-Muller code is used, which can be completely described by a single parameter $m$. This parameter defines the length of the information word $(m + 1)$ bits, the length of the codewords $2^m$ bits, the generator matrix $G$, as well as all other necessary variables. An empirical analysis of the Reed-Muller ECC will be presented in Section 5.8.

First of all, the codewords generated by Reed-Muller have a length of $2^m$ bits for a first-order Reed-Muller code $\mathcal{R}(1, m)$. The closest codeword lengths for the desired Bloom filter length $l = 23905$ are achieved using $m_1 = 14$ and $m_2 = 15$, which yield a bit string length (for the Bloom filter and codeword) of 16384 bits or 32768 bits, respectively. As will be shown later, the ECC works best when the normalized Hamming weight $\bar{w}(b)$, which is the Hamming weight of a bit string $b$ divided by its length $\bar{w}(b) = {}^{w_{\mathrm{H}}(b)}/_{\|b\|}$, is close to 0.5 ($\|b\|$ specifies the length of the

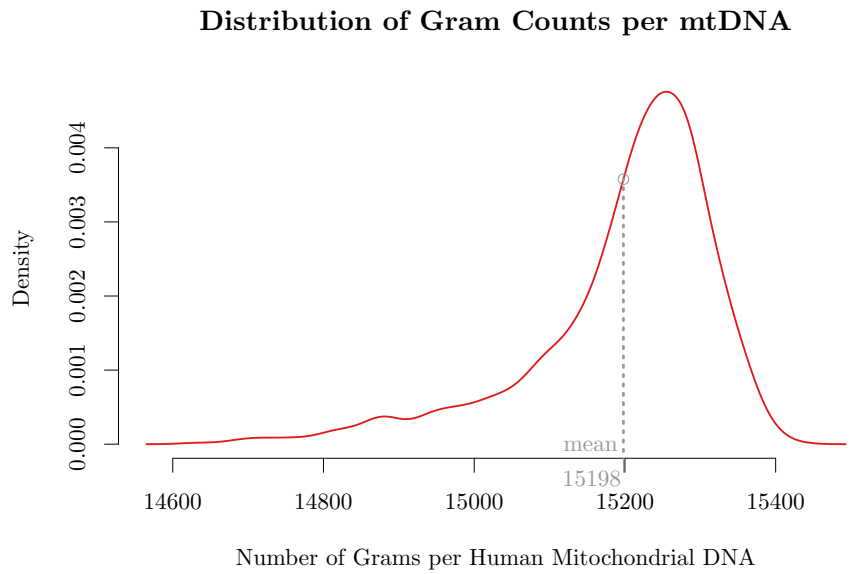



Figure 5.7.: Distribution of the number of grams generated by the VGRAM algorithm [LWY07] for mitochondrial DNA sequences taken from the genome database [IG06].

vector $b$). The Hamming weight of the bit string $b$ (the Bloom filter) is determined by its length and the number of inserted elements.

The number of inserted elements into the Bloom filter equals the number of variable length grams generated for the input string at the corresponding party. In previous sections the number of grams generated was approximated to be close to the actual input string length, which is 16568. Figure 5.7 shows the distribution of the number of variable length grams generated for DNA sequences from the human mtDNA database [IG06]. The mean number of grams for a human mitochondrial genome sequence is 15198. As the grams are input to the Bloom filter, the number of elements inserted into a Bloom filter equals the number of grams. Following this, the average number of inserted Bloom filter elements is equally 15198.

Under the restriction of using a single Reed-Muller code, the only available Bloom filter sizes are $2^{14}$ bits or $2^{15}$ bits. Equation (4.1) estimates the number of set bits in a Bloom filter $b$ given the number $i$ of inserted elements and the length $l$ of the Bloom filter. Dividing the expected number of set bits by the length of the filter results in the approximated normalized Hamming weight $\bar{w}(b) = 1 - \exp(-i/l)$ (see Equation (4.1)). Using $i = 15198$, the expected normalized Hamming weights are either $\bar{w}_{14}(b) = 0.604$ or $\bar{w}_{15}(b) = 0.371$. Both values are too far off from 0.5 to be applicable. Following these constraints, a Bloom filter of size $2^{14}$ bits

or $2^{15}$ bits does not produce the desired normalized Hamming weight as given by Equation (4.1) in Section 4.2. It is concluded that a single, large Reed-Muller code is unsuited for the desired functionality and we thus use several shorter ones to be able to set the bit string length more fine-granular. As the total number of elements that are inserted into the Bloom filter is fixed, two variables remain flexible. That is the length of the Bloom filter and the normalized Hamming weight of it. Therefore we are going to analyze the ECC decoding equations analytically for different ECC lengths and study the behavior of the ECC decoding function depending on bit strings with different normalized Hamming weight.

As a next step we analyze the decoding algorithm of a first-order Reed-Muller code to learn what settings are suitable to decode similar Bloom filters to the same information word and dissimilar strings into different ones. The standard Reed-Muller decoding algorithm describes for each but one bit of the information word a set of $2^{m-1}$ decoding equations, each summing up exactly 2 distinct bits of the Bloom filter. The final bit of the information word is generated using a majority vote over the result of all decoding equations out of the respective set. If there are equally many zeroes and ones, a decoding failure occurs. See Section 3.7 for a detailed description. To circumvent this situation, we propose to use a shortened Reed-Muller code, which skips one of the decoding equations (the last one) to have an odd number of votes and thus never reach a draw during voting.

#### Reed-Muller Analysis

To analyze the Reed-Muller decoding, we are going to analyze the probability of two random bit strings $b_1, b_2$, having Hamming distance $k$ and a length of $2n = 2^m$ bits to decode into different information words. The first of the two strings is a random bit string $b_1$, while the second bit string $b_2$ is a copy of the first, which is changed at $k$ different, random positions, chosen uniformly. Let $b$ be a placeholder of either of the two.

Following Section 3.7, decoding $b$ implies evaluating $2^{m-1}$ linear equations for every bit (except the last one) of the information word. The equations are XORs ($\oplus$) between two different bits of $b$. To simplify the following analysis, we arrange all bits from $b$ in two lists of length $b/2$ in such a way that the decoding equations always use the same index for both lists during decoding. This is achieved by applying a permutation $\pi_i$ on $b$ as a model for selecting the appropriate bits for decoding. The permutation differs for each of the herewith decoded information word bits, however all $m$ permutations are fixed for $\mathcal{R}(1, m)$. The resulting bit string is split in two half's of the same length $\pi_i(b) = b_L^i | b_R^i$, with "|" denoting a concatenation of bit strings. $b_L^i$ denotes the first $n$ selected bits by the permutation $\pi_i$ and $b_R^i$ denotes the last $n$ bits. The permutation is chosen such that for a fixed

information word bit at position $i$, the decoding equation $eq_j^i$ takes the form $eq_j^i = b_L^i[j] \oplus b_R^i[j]$. Let $b_\oplus^i = b_L^i \oplus b_R^i$ denote the result of evaluating and concatenating all $n$ decoding equations for the $i$-th information word bit. Decoding of an information word bit at position $i$ is then described as a majority vote over $b_\oplus^i$. Recall that we use modified decoding, *i.e.* we use an odd number of decoding equations, which means that the length of $b_\oplus^i$ is also odd. $b_l^i(b_R^i)$ is the vector of length $n$, which takes all elements denoted by 0(1) in Figure 3.3, therein termed vector $r_i$ $(1 \leq i \leq m)$.

The following analysis focuses on the probability of changing the outcome of decoding a single information bit. "Changed outcome of decoding" refers to the result of the majority vote over $b_\oplus$, *i.e.* the vote returns 1 for decoding a specific bit of $b_1$, but returns 0 decoding the same information word bit on $b_2$ — or vice versa.

**Modeling $\mathcal{R}(1,m)$ Decoding** Let $b_1, b_2$ be two compared bit strings of identical length and Hamming distance $k$. We fix the information word bit that is to be decoded and don't change it unless specifically mentioned. Decoding the selected information word bit is simulated by generating $b_\oplus$ using all $n$ decoding equations. Let $n_0$ denote the number of zeroes and $n_1$ the number of ones in $b_\oplus$. Furthermore, the total length of $b_\oplus$ is $n = n_0 + n_1$ with $n$ being odd due to the use of a shortened Reed-Muller code. The decoding algorithm — more precisely: the chance to decode a fixed information word bit differently in both bit strings $b_1$ and $b_2$ — is modeled as a two step experiment. For ease of presentation and without loss of generality, we assume that $n_1 > n_0$ for $b_\oplus$ on bit string $b_1$, which simply means that the decoded information word bit is 1 using a majority vote over $b_\oplus^{(1)}$. We then derive the probability $\bar{p}_{\text{bit}}$ of reaching $n_0 > n_1$ for $b_2$ —*i.e.* decoding the selected information word bit to 0 instead of 1 for the other compared bit string. In cases where we need to distinguish between $b_\oplus, b_L, b_R, n_0, n_1$ for both compared bit strings $b_1, b_2$ we add $^{(1)}$ and $^{(2)}$ as in $b_\oplus^{(1)}$ for $b_1$ or $n_0^{(2)}$ for $b_2$. The opposite case, that $n_0 > n_1$ for $b_1$ can be reduced to the aforementioned case $n_1 > n_0$ by flipping all bits in $b_L$.

As we are interested in the probability $\bar{p}_{\text{bit}}$ of decoding a single information word bit differently between a given $b_1$ and a random $b_2$ with Hamming distance $k$ towards $b_1$, we need to split the problem of calculating the desired probability into smaller problems that can be tackled easily. From the assumption $n_1 > n_0$ follows that a different decoding happens in case of $n_0 > n_1$. Thus, we need to calculate the probability for that case to happen given $b_1$ and $k$ as input. As $n = n_0 + n_1$, we can reformulate the inequality as $n_0 > n/2$. We therefore have $\bar{p}_{\text{bit}} = \Pr[X > n/2]$. The random variable $X$ denotes the number of 0's in $b_\oplus^{(2)}$ after all $k$ bit flips. Using

the law of total probability, we break up the probability for the inequality into a sum of probabilities for concrete events as given in Equation (5.3).

$$\Pr[X > n/2] = \sum_{m=\lceil n/2 \rceil}^{n} \Pr[X = m] \tag{5.3}$$

We described the experiment as a two step process. In the first step of the experiment we draw $l \leq k$ positions from $b_L^{(1)}$ of $b_1$ and flip the bits at the corresponding positions to construct a $b_L^{(2)}$. In the second step we flip $k - l$ bits at random positions in $b_R^{(1)}$ to generate $b_R^{(2)}$ and calculate the probability that these second bit flips together with the first $l$ flips generate a difference $i = n_0^{(2)} - n_0^{(1)}$ between $b_\oplus^{(1)}$ and $b_\oplus^{(2)}$.

We start with considering the probability of having a Hamming distance of $l$ between $b_L^{(1)}$ and $b_L^{(2)}$ given a Hamming distance of $k$ between $b_1$ and $b_2$. Recall that the number of draws in the first step is $l$ over a string of bit length $n$. The number of total draws is $k$ over a bit string of length $2n$. Therefore the probability of having $l$ draws in the first $n$ bits is equivalent to the probability given by the hypergeometric distribution, which describes the probability of having $l$ successful draws out of a total of $k$ draws with in total $n$ possible successes and a total population of $2n$. Equation (5.4) uses the probability mass function of the hypergeometric distribution to calculate the probability for the event that the random variable $F_1$, which denotes the number of flipped bits in $b_L$, equals $l$.

$$\Pr[F_1 = l] = \frac{\binom{n}{l} \cdot \binom{n}{k-l}}{\binom{2n}{k}} \tag{5.4}$$

Given the probability of having exactly $l$ bits flipped in $b_L$ we need to calculate the probability that these $l$ bit flips changed the number of 0's between $b_\oplus^{(1)}$ and $b_\oplus^{(2)}$ by $i$: $i = n_0^{(2)} - n_0^{(1)}$. The random variable $\tilde{X}$ denotes the number of zeros in $b_\oplus^{(2)}$ after flipping the first $l$ bits. Due to flipping $l$ bits upon a bit string, the possible values for $i$ in the change of 0-bits between the original string $b_\oplus^{(1)}$ and the generated string $b_\oplus^{(2)}$ have a specific structure $i \in \{-l, -l+2, \ldots, l-2, l\}$. If $l = 1$, then either a 1 in $b_\oplus^{(1)}$ is changed to a 0 in $b_\oplus^{(2)}$, resulting in $i = 1$, or a 1 is changed to a 0, resulting in $i = -1$. Similarly for $l = 2$ follows $i \in \{-2, 0, 2\}$, for $l = 3$ is $i \in \{-3, -1, 1, 3\}$ and so forth. Given this structure and a helper variable $a_0^1 = \frac{l-i}{2}$, it holds that $a_0^1 \in \{0, \ldots, l\}$. The variable $a_0^1$ describes the number of bits in $b_\oplus^{(1)}$ which were changed from 0 to 1 in $b_\oplus^{(2)}$. At this point we can again use a hypergeometric

distribution to describe the probability distribution for having exactly $i$ changed 0's between $b_{\oplus}^{(1)}$ and $b_{\oplus}^{(2)}$. The problem is reduced to hypergeometric distributions by letting $a_0^1$ — *i.e.* the number of 0's changed to 1's — be the number of successes out of $l$ total draws with a maximum number of successes $n_0^{(1)}$ and a population count of $n$ bits to choose from in total. Equation (5.5) uses the probability mass function of the hypergeometric distribution to describe the probability of a concrete event that changes the number of 0's by $i$ in $b_{\oplus}$ given $l$ bit flips in the first $n$ bits. The probability for the event that exactly $l$ out of $k$ bits are flipped in the first $n$ bits was given by Equation (5.4). As the probability for $i$ changes given a specific $l$ is only defined for a certain $i$ as given above, Equation (5.5) includes a case for $a_0^1 \notin \{0, \ldots, l\}$ returning a probability of 0.

$$\Pr[\tilde{X} = n_0 + i | F_1 = l] = \begin{cases} 0 & \text{if } a_0^1 \notin \{0, \ldots, l\} \\ \frac{\binom{n_0}{a_0^1} \cdot \binom{n-n_0}{l-a_0^1}}{\binom{n}{l}} & \text{else} \end{cases} \tag{5.5}$$

The second step in our experiment is to draw the remaining $k - l$ positions from the $n$ bits in $b_R^{(1)}$, flip them, construct $b_R^{(2)}$ and calculate the final probability that $\Pr[X = m]$, as required for our "different bit decoding" probability. We use the same approach as before by applying the probability mass function of the hypergeometric distribution upon a new helper variable $a_0^2 = \frac{n_0+i+k-l-m}{2}$, which describes the number of 0's in $b_{\oplus}$ that are changed by the second experiment step. This also includes the zeroes generated previously by the first experiment step and therefore includes the case that the XOR-operation executed by the decoding equations might cancel out some of the $k$ bit flips on the original bit string $b_1$. Equation (5.6) describes the probability of reaching $m$ 0's in $b_{\oplus}^{(2)}$ after applying all $k$ bit flips, given that $l$ flips happen on the first $n$ bits and $i$ zeroes are changed after the first $l$ flips. Similarly to Equation (5.5), a special case takes care of values for $a_0^2$ that are undefined.

$$\Pr[X = m | \tilde{X} = n_0 + i, F_1 = l] = \begin{cases} 0 & \text{if } a_0^2 \notin \{0, \ldots, k-l\} \\ \frac{\binom{n_0+i}{a_0^2} \cdot \binom{n-n_0-i}{k-l-a_0^2}}{\binom{n}{k-l}} & \text{else} \end{cases} \tag{5.6}$$

We can put all the single probabilities together and sum over all the introduced variables to reach a combined probability for $\Pr[X = m]$, as given in Equation (5.7).

$$\Pr[X = m] = \sum_{l=0}^{k} \sum_{i=-l}^{l} \Pr[X = m | \tilde{X} = n_0 + i, F_1 = l] \cdot \Pr[\tilde{X} = n_0 + i | F_1 = l] \cdot \Pr[F_1 = l] \tag{5.7}$$

We have defined how to calculate the probability that a certain decoded information bit changes if we apply $k$ bit flips at random positions to a given bit string $b_1$. In the following sections we combine this result to derive a probability for a changed information word and finally a changed Bloom filter decoding, or fuzzy commitment, as necessary to detect similar queries for inference attack purposes.

### Reed-Muller Analysis Results

As the decoding algorithm (see Section 3.7) actually describes $m$ sets of decoding equations, the probability of a single change throughout all of these bit decodings is required. Or in other words, if a single bit decodes differently between $b_1$ and the $k$ bits distant other bit string $b_2$, than both produce different fuzzy commitments and are not identified as similar inputs. We are thus interested in the probability of having any of the decoded bits changed. As there are $m + 1$ decoded information bits, we thus calculate the probability of having $b_1$ and $b_2$ decode into different information words:

$$\bar{p}_{\text{code}} = 1 - (1 - \bar{p}_{\text{bit}})^{m+1}.$$

Furthermore, next to $m$ sets of terms or relations for majority decoding of $m$ bits, the last bit is computed using a simple majority decoding over a reduced bit sequence $b'$, as specified in Section 3.7. This step is a simplification as it assumes that all decoded information bits are independent, which they aren't. However, we will later on see that this is a good approximation.

As a single Reed-Muller ECC was too restrictive on its possible length, $r$ shorter codes are used to get closer to the desired bit string length. For example, using $m = 10$ allows the bit string length to be a multiple of $2^{10}$. The overall probability of decoding two Bloom filters into different commitments is therefore given by $\bar{p}_{\text{total}}$.

$$\bar{p}_{\text{total}} = 1 - (1 - \bar{p}_{\text{code}})^{r}$$

Figure 5.8 gives an overview on how the probability of decoding into different information words ($p_{\text{total}}$) evolves for two bit strings with different Hamming distance $k$ and one of the bit strings having a certain Hamming weight offset $o$ from $2^{m-1}$. Let $s_0$ be a bit sequence of length $n$ with a Hamming weight $w_{\text{H}}(s_0) = \lfloor n/2 \rfloor$, then

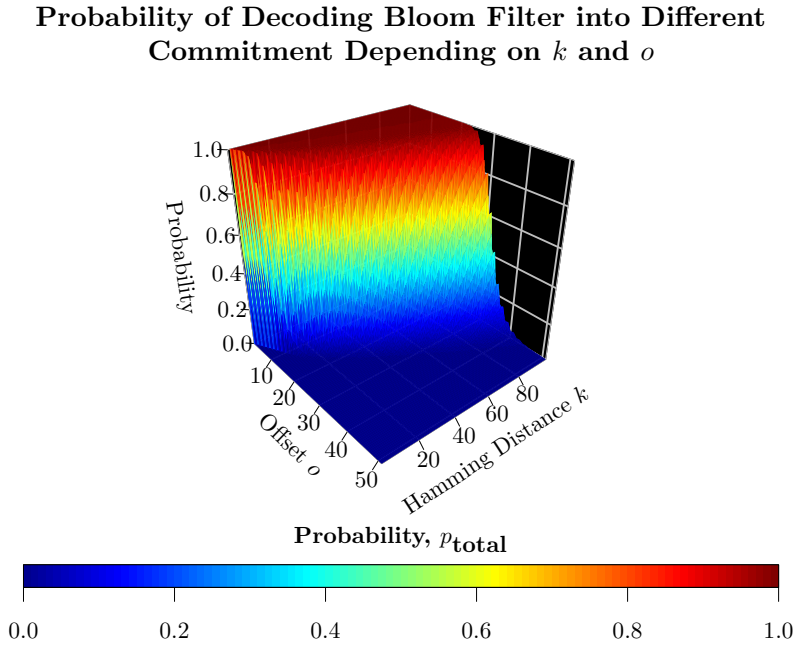



Figure 5.8.: Probabilities to decode into a different commitment depending on codeword offset $o$ and Hamming distance $k$ between Bloom filters. The plotted probabilities are values obtained for $p_{\text{total}}$ as described in the theoretical Reed-Muller analysis.

a bit sequence $b$ has offset $o = w_{\text{H}}(b) - w_{\text{H}}(s_0)$. Put differently, this offset determines the Hamming distance between the bit string $b_1$ and a word with Hamming distance $\lfloor n/2 \rfloor$. $b_1$ and a Hamming distance $k$ is used as input in the theoretical analysis above.

The blue plane at the bottom of the graph represents the area in which the parameters Hamming weight of Bloom filter and Hamming distance to another Bloom filter can vary, without decoding into a different commitment. However, as we move to the red plane, that is the region of higher distances between Bloom filters or lower offsets (Hamming weights close to 0.5), the probability of decoding into a different information word is very close to one. The blue and red areas are connected via a very steep upward curve between them. It can be seen that by choosing the offset — that is the normalized Hamming weight — of the Bloom filters properly, the detection of similar Bloom filters, by the definition of Hamming distance, can be steered nicely.

Figure 5.8, does not show the probability distribution for a single ECC word, but for the whole Bloom filter to decode into the same commitment. The adaptation

of the probability for a differently decoded word towards a differently decoded total Bloom filter is straightforward, as the bits used for the next ECC word can be assumed to be independent (they are set using a cryptographic hash function, which is known to be a good randomness extractor and thus generate output close to the uniform distribution). It takes the inverse probability of a change — the probability to decode identically for a single word — raised to the power of $r$ (to get to the length of the Bloom filter) and subtracted from one again. In the discussed case $r = r_{\max} = 22$.

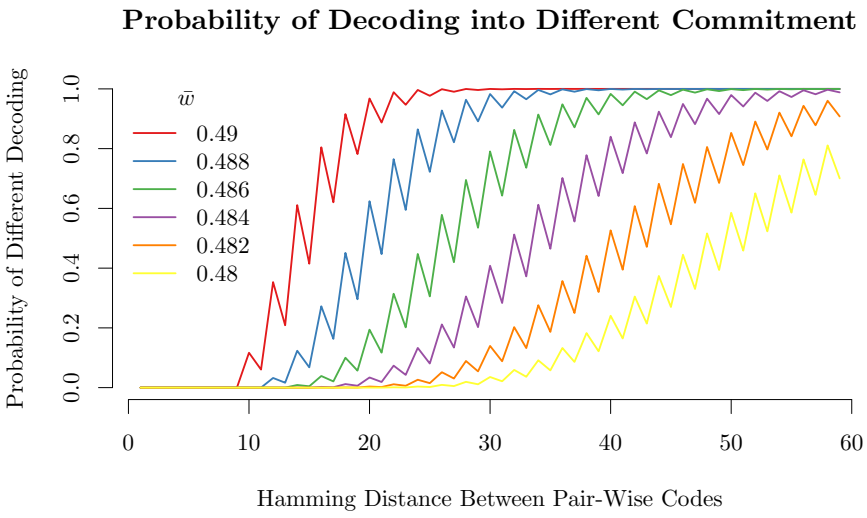



Figure 5.9.: Probabilities to decode two bit strings into different information words depending on normalized Hamming weight and Hamming distance to another Bloom filter.

For example it follows for a bit string $b$ of length $2^m \cdot r = 22528$ bits ($m = 10, r = 22$) and normalized Hamming weight $\bar{w}(b) = 0.486$ (the green curve in Figure 5.9) that another string with a Hamming distance of up to 16 bits in all $r$ single ECCs substrings of length $2^m$ that are used for decoding, will be decoded into the same commitment with high probability. Figure 5.9 highlights that case for different normalized Hamming weights $\bar{w}(b)$ and Hamming distances between the two bit strings.

At last, it is to note that once a desired normalized Hamming weight is picked, *i.e.* $\bar{w}(b) = 0.486$, Equation (3.2) is used to derive the correct Bloom filter length, which provides normalized Hamming weights close to the desired value $p_{\text{fp}} = \bar{w}(b)$. The calculation is based on the average number of grams generated by the VGRAM algorithm. In our example the calculated Bloom filter length would be 22836. As the length clearly differs from the possible length of $2^m \cdot 22 = 22528$ ($m = 10$),

it must be specified how to map 22836 bits (size of the Bloom filter) to 22528 bits (total size of the $r$ ECC words). Section 5.8 will discuss, implement and evaluate three such mappings. One possibility — that will also be evaluated later on — is to insert all grams into the full length Bloom filter and remove the last bits of the filter to reduce the filter size to a multiple of the selected ECC length. This essentially resembles a sampling of the Bloom filter and in the specific case only a very small fraction is removed ($< 1.4\%$). This procedure is identical to Locality-Sensitive Hashing (LSH)-families that perform bit sampling in Hamming spaces, as done by Gionis, Indyk, and Motwani [GIM99] and Indyk and Motwani [IM98], and offers a very good approximation especially at this high sampling rate. Despite not drawing or removing bits from random positions or dimensions, the bit strings themselves are treated as random strings, making the random index choice for sampling unnecessary. Furthermore, the high accurate sampling can be used to select the average normalized Hamming weight very precisely, as arbitrary sized Bloom filters can be used to exactly mimic the desired normalized Hamming weight. Sampling the Bloom filter then brings its size down to the actual required size.

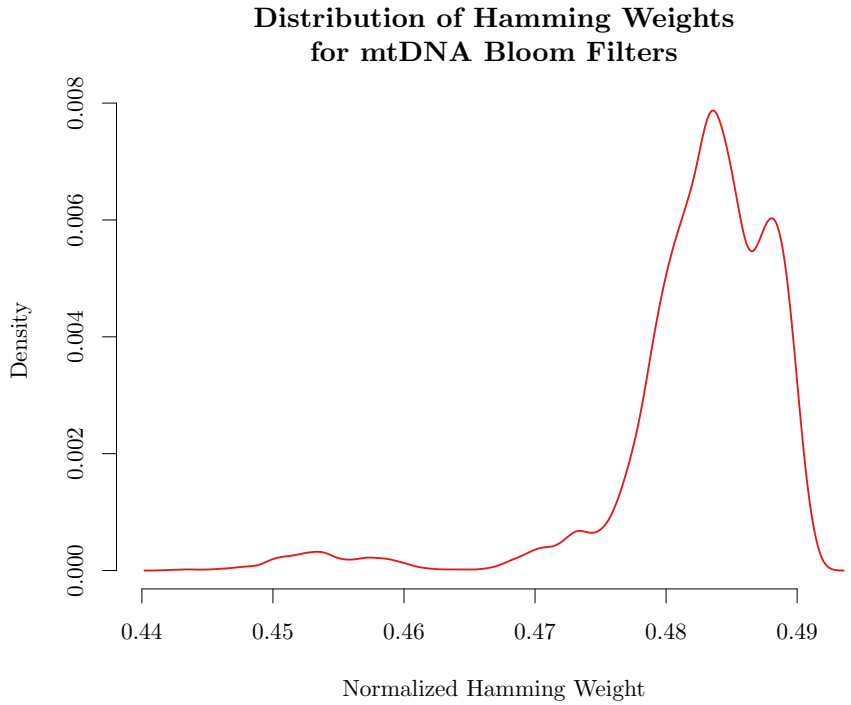



Figure 5.10.: Distribution of Hamming weights for all Bloom filters generated over the DNA database [IG06]. Approximately 78% of all normalized Hamming weights are above 0.48, within the desired range below the desired normalized Hamming weight of 0.5, as shown in Figure 5.9.

The actually achieved distribution of normalized weights over the database is presented in Figure 5.10 and nicely shows that the interesting range of normalized Hamming weights $[0.48, 0.49]$ includes over 75% of all database entries, with the median being $\bar{w}(b) \approx 0.484$.

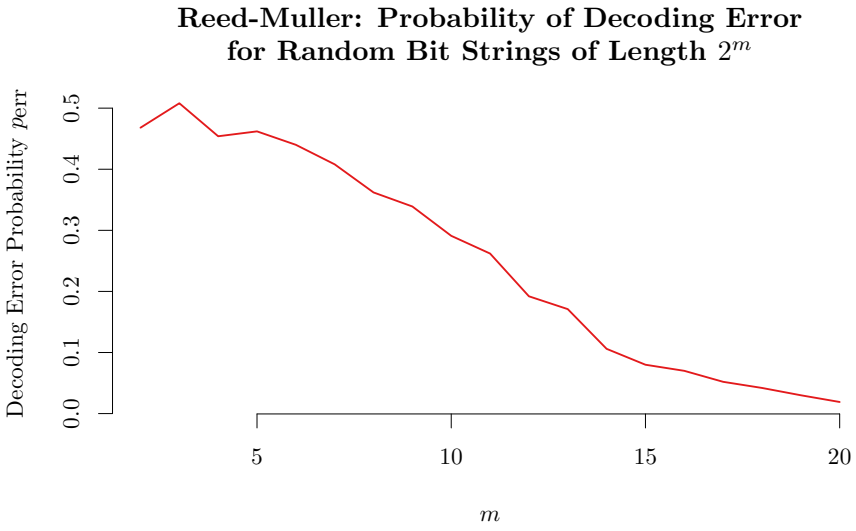



Figure 5.11.: Probability of a decoding error for random bit strings using a Reed-Muller code with $m \in [2, 20]$. The error probability $p_{\text{err}}$ decreases fast with an increasing $m$. At $m = 15$ the probability of a decoding error is already below 10%: $p_{\text{err}} = 0.08$

This plot together with Figure 5.11 helps in deciding which value for $m$ to choose. In the appendices (Section B.1) follows an overview on the influence of the choice of $m$. The requirements are that the probability of a decoding error for a random bit string (Figure 5.11) should be low to allow a few bits to be changed and still decode to the same information word and finally into the same commitment. In the end we want to detect inference attacks by means of mapping into the same information word. Furthermore, we want a high number of "Message Information Bits" from Figure B.1, as these specify the size of the domain that can be successfully compared without being detected as similar values. Similarly to the low probability of receiving a decoding error for a random string, the actual ECC should be able to correct a possibly large amount of bit flips, denoted as "Single ECC Correctable Bits" in the Reed-Muller configuration comparison figure. From all of this it follows that $m = 10$ seems to be a reasonable choice as a compromise between being able to correct errors (Hamming distances between two genuine genome Bloom filters) and the size of the domain that can be compared. Values for both parameters are most similar for $m = 10$. We therefore continue our analysis with this choice.

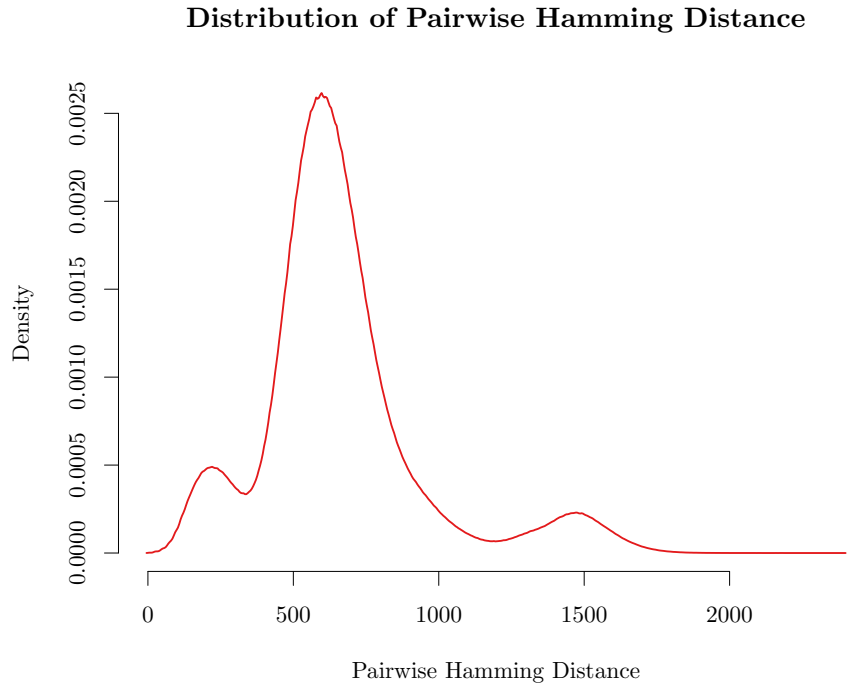



Figure 5.12.: Distribution of pair-wise Hamming distance between all Bloom filters generated out of the "Human Ditochondrial Genome Database" [IG06] using the scheme presented in Chapter 4 and a Bloom filter length of 22836 bits.

The distribution of actual Hamming distances found in the database of Ingman and Gyllensten [IG06] — as plotted in Figure 5.12 — demonstrates that it is very likely that a pair of genuine genomic sequences, as they are found in the database, has a Hamming distance above 500 using our measure. Furthermore, as 22 ECCs, each with a length of $2^{10}$ bits are used to decode the Bloom filters, all ECCs have to handle on average $\approx 22.7$ bit flips. Figure 5.9 shows that, especially for Bloom filters $b$ with normalized weights $\bar{w}_m(b)$ close to 0.5, such an amount of flipped bits very likely results in the overall decoded message being changed and thus mapped to a different commitment. This is even more true when the overall changed bits — the Hamming distance $d_{\mathrm{H}}$ between two Bloom filters is viewed as one of the Bloom filters being flipped at $d_{\mathrm{H}}$ different positions — are not distributed equally over all the 22 ECCs. The probability of a single ECC decoding differently raises when it receives more flips and so does the probability for the overall mapping to another commitment.

## 5.8. Empirical Evaluation

Next to the analytic evaluation of our proposed inference attack detection scheme in Section 5.7.4, we present the results obtained from evaluating our scheme over all unique genomes in the used database [IG06] under selected configurations. To better understand the details of the empirical evaluation, we quickly recall the overall concept and comment on important details. As described in the construction in Section 5.4, each genomic sequence is mapped to a set of variable length grams using the VGRAM algorithm [LWY07] and all sets of a sequence are inserted into a Bloom filter. These Bloom filters are then decoded using the selected ECC (shortened Reed-Muller) with a configuration derived from our analytic results in Section 5.7.4.

We use a Bloom filter length of $l = 22836$ bits and utilize the filters to store on average 15198 grams per genomic sequence in each filter. Using this configuration, we reach a normalized Hamming weight in the range $[0.48, 0.5]$ for about 78% of the Bloom filters build from the genomic database as depicted in Figure 5.10. A normalized Hamming weight in this range was found to deliver the desired performance. Furthermore, the configuration of the ECCs was set to $m = 10$, which balances the domain size for the possible genomes with the error-correcting capability of the ECC. As a result the ECC was combined $r = \lfloor l/2^m \rfloor = 22$ times to reach a combined reduced length of $l_r = 2^{10} \cdot 22 = 22528$ bits. All Bloom filters were thus shortened by removing $l - l_r = 308$ bits to fit the reduced length of the combined ECC size boundary.

Until now we did not mention how the Bloom filter bits are mapped to ECC words. Therefore we will describe and discuss different mappings.

In general we propose to map bit positions of the pseudo-random bit strings (Bloom filters) to positions within the words that are to be decoded by the ECC. This map can be implemented in three steps:

1. Permute the bits of a Bloom filter using permutation $\pi$
2. Remove the last $l - l_r$ bits
3. Select $2^m$ consecutive bits for all $r$ ECC words

These three steps are also depicted in Figure 5.13 together with the conceptually following decoding step. We will also describe three different ideas to construct the permutation $\pi$ and evaluate the respective influence on the detection capability. Those three techniques are:

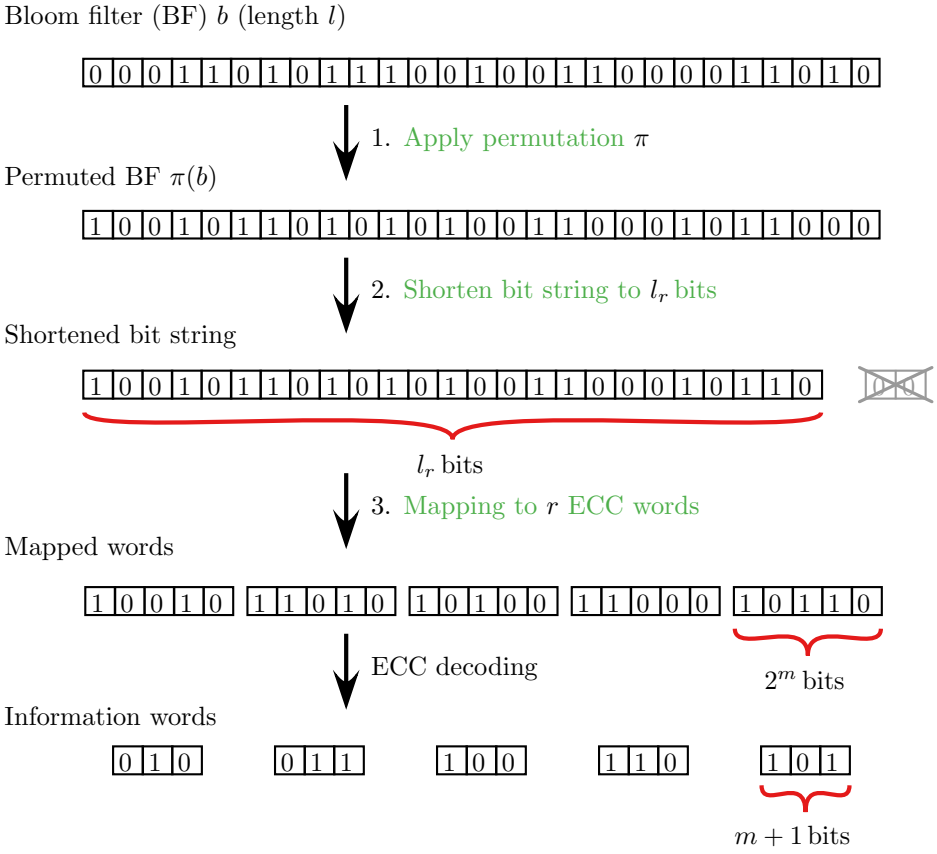



Figure 5.13.: Steps performed to map a Bloom filter to a bit string that will be decoded using an ECC. The supplied word bit lengths are specific for the selected Reed-Muller ECC.

1. No permutation (Sequential)
2. Normalized Hamming weight of 0.5 (Uniform)
3. Close to codeword (Codeword)

However, before mapping strategies are described, a generic requirement must be ensured. The final domain to which all compared elements will be mapped to must be large enough to actually represent all compared elements with high probability. The size of the target domain actually equals the size of the domain of information words (of the ECC) multiplied by the number of information words used to represent a single genomic sequence or in general an element that should be used for comparison. After decoding, each ECC returns $m + 1 = 11$ bits as an information word, which results in a total of $11 \cdot 22 = 242$ bits to represent each genomic sequence. We can thus at a maximum distinguish between $2^{242}$ different genomic sequences, or following the birthday paradox [GB08] approximately $\sqrt{2 \cdot 2^{242}} \approx 2^{121}$ different sequences, which should still be enough in practice.

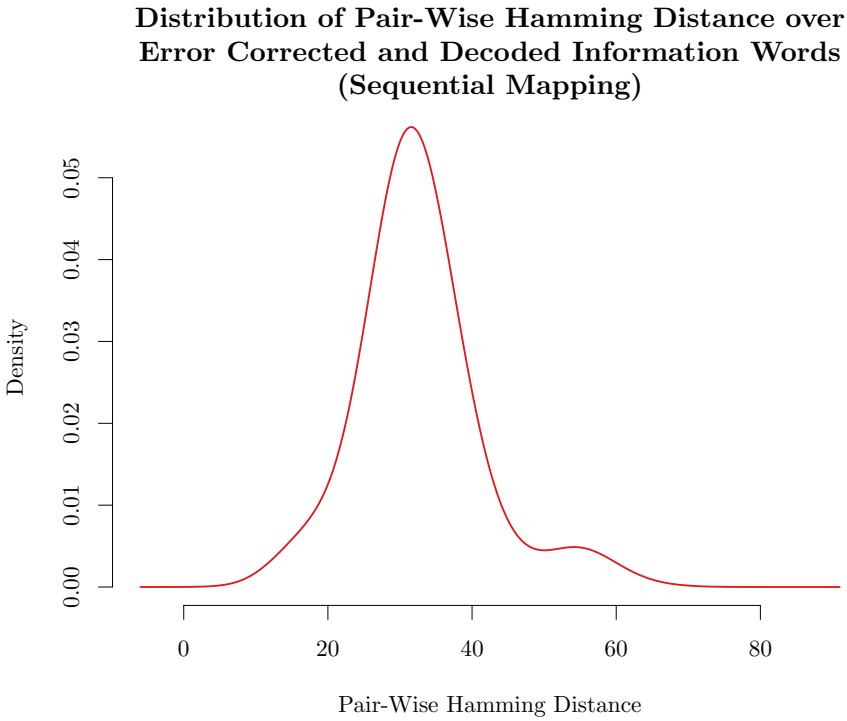



Figure 5.14.: Distribution of Hamming distances for combinations of decoded information words out of the "Human Mitochondiral genome Database" [LWY07].

**1. No permutation** We start with the intuitive mapping, which takes the first $2^m$ bits of the Bloom filter and decodes them to retrieve an information word, then the next $2^m$ bits for the second information word and so on for all $r = 22$ ECC invocations. So this mapping applies the identity permutation (or no permutation) and uses the first $2^m \cdot r$ bits of the Bloom filter. All Bloom filters of the genome database were decoded this way and all possible combinations of decoded information words were then compared. Figure 5.14 shows the pair-wise Hamming distance distribution of an all-to-all comparison over the received information words. Recall that if within these combinations the same information word appears, it implies that also both related genomic sequences are mapped to the same fuzzy commitment and thus the query for the second of these two genomes would be rejected as possible inference attack. This is equivalent to a Hamming distance of 0 between any of the information words.

The analysis for the probability of decoding into a different combined information word over the whole Bloom filter $b$ given a certain normalized Hamming weight $\bar{w}(b)$ and a number of flipped bits (Hamming distance to the compared Bloom filter), assumes that all used ECCs have the same normalized Hamming weight.

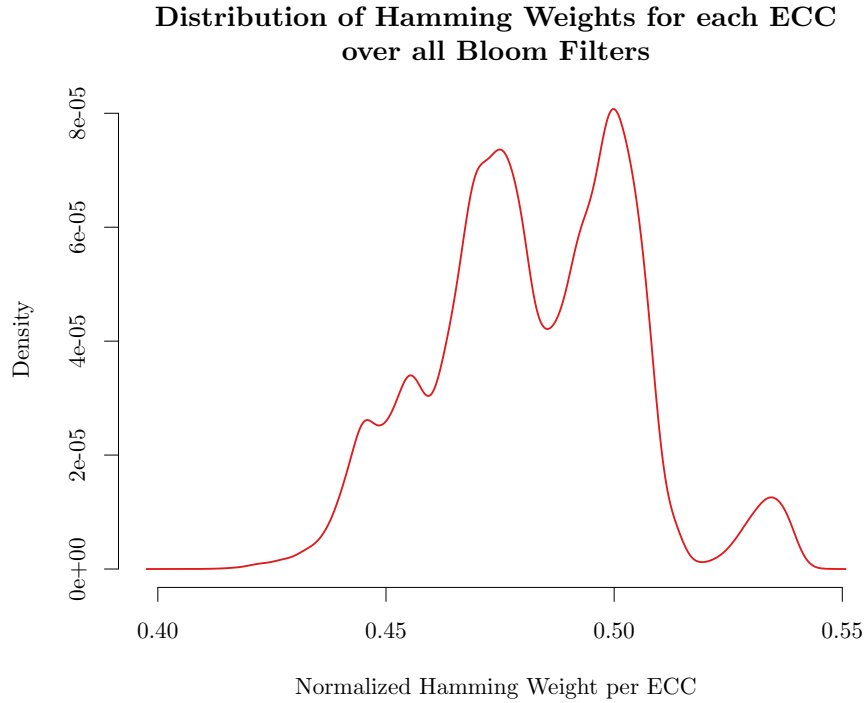



Figure 5.15.: Distribution of normalized Hamming weights $\bar{w}(b)$ per Error-Correcting Code for all Bloom filters generated over the whole database. The indices used for the Error-Correcting Codes are selected sequentially over the Bloom filter.

This might be correct on average for truly random strings, but it will most certainly not be true for a single use case. This use case follows from converting genomic sequences into Bloom filters. As described above, assume that the first ECC out of the 22 used for a single Bloom filter uses all bits with an index in the range $[1, 2^m]$, the second ECC uses the directly follow bits with indices $[2^m + 1, 2 \cdot 2^m]$ and so on. So the ECCs are dividing the Bloom filter into 22 equally sized parts of sequential bits. The distribution of normalized Hamming weights of the ECC words for all Bloom filters of the complete database using such a sequential mapping from Bloom filter bits to ECCs words is depicted in Figure 5.15.

One can easily see that the distribution distinctly differs from the desired one depicted in Figure 5.10, which shows the distribution of normalized Hamming weights over the complete Bloom filters. Furthermore, Figure 5.15 shows that a large number of bit strings exhibit a normalized Hamming weight closely around 0.5, which means that already a slight change on the string (a single bit flip) will very likely change the result of the decoding, as indicated by Figure 5.9. As a result, most slight changes made to a request to compare genomic sequences will

result in at least one of the ECCs decoding to another information word and thus to another fuzzy commitment. We would see a very low number of false-positives in comparing genuine genomic sequences, but on the other side nearly all inference attack queries would go undetected — even if only a small number of bits would be different between the inference attack query and a previously genuine query.

If this setting is applied to the used database, utilizing the parameters mentioned above and a sequential selection of the ECC bits, then the total number of false-positives found via a complete all-to-all comparison is 850. Therefore — there are 12064 unique genomic sequences in the mtDNA database, yielding a total of $\binom{12064}{2} = 72764016$ overall pair-wise combinations — approximately 99.999% of all these comparisons will succeed without raising an inference attack alarm. In other words, the false-positive rate of detecting two genuine mitochondrial genomic sequences as an inference attack is below 0.000012. However, detecting inference attacks also supports only a few changed bits. In other words, changing a single character in the genomic sequence changes more grams and thus more bits in the Bloom filter than the ECC is capable of identifying as identical, which leads to virtually no detection capability when changing the genomic sequence — even by a single character.

**2. Normalized Hamming weight of 0.5** To solve the before mentioned issue and increase the inference detection capability, we must ensure a better error-correction capability of the ECC given our constraints. Looking at the results of the previous (sequential) mapping, it strikes that the normalized Hamming weight distributions of the Bloom filters and that of the derived ECC words (compare Figure 5.15 and 5.10) are very different. An improved second mapping therefore tries to match the normalized Hamming weight ECC word distribution to the Bloom filter distribution of normalized Hamming weights. In other words, a mapping should be found which results in a more uniform bit distribution for the ECC words.

To solve this issue, the bits for each ECC are not selected sequentially, but in such a way that we get a more similar normalized Hamming weight distribution. Of course the actual mapping of bits from Bloom filters to ECCs words must be fixed ahead of the first comparison and be identical for all queries that should be comparable. If the mapping — or actually the permutation — would be chosen based on the compared sequences, the resulting information words and thus also commitments (see Section 5.4.2) could not be compared and furthermore, the description of the permutation would leak information about the input sequence(s).

The issue of finding a good permutation or index selection which is independent on the actual private input is dealt with bye making use of the property that all input sequences and thus also their binary Bloom filter representations are very similar in general. Therefore, the bit distribution of a publicly known reference

sequence, the "revised Cambridge Reference Sequence" rCRS is used to derive a permutation or mapping from bits of the Bloom filter to the ECCs words.

It selects $2^m = 1024$ random bits from the Bloom filter generated out of the rCRS and checks if its normalized Hamming weight is close to the overall normalized Hamming weight of the reference Bloom filter. If this is not the case random bits are sampled again. *Close to* refers to a distance that can be chosen depending on the actual context of the comparison. In this thesis a threshold of 0.5% between both normalized weights was used. This number was chosen, as it narrows the possible pair-wise difference between the normalized weights of the single ECCs for that Bloom filter down well enough (compare with Figure 5.9) and it is still possible to actually find such a mapping over the rCRS Bloom filter for all $r = 22$ ECC words.

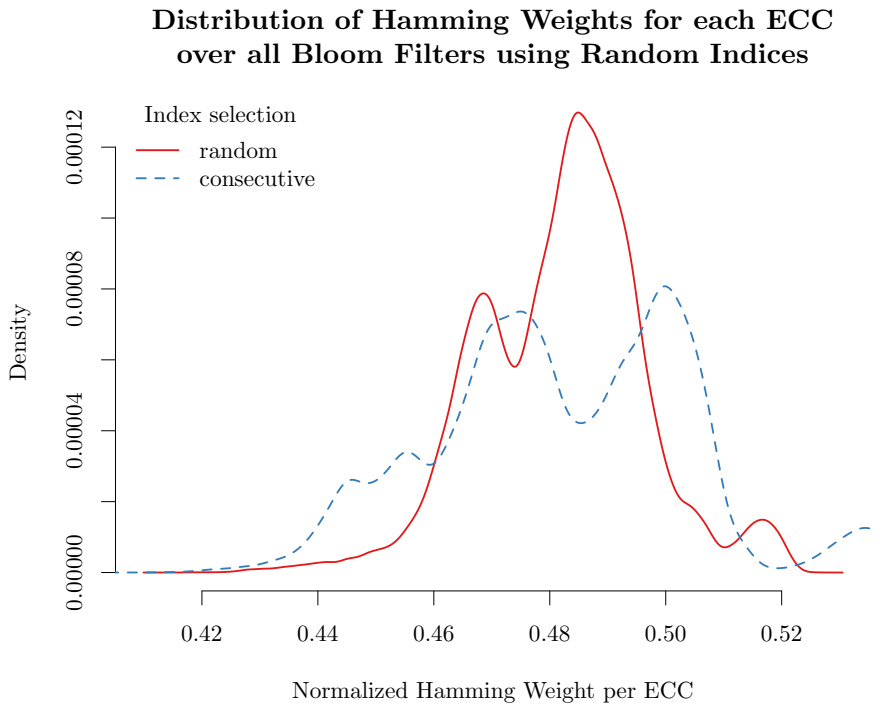



Figure 5.16.: Distribution of normalized Hamming weights $\bar{w}(b)$ per Error-Correcting Code for all Bloom filters generated over the whole database. The indices used for the Error-Correcting Codes are selected randomly over the rCRS Bloom filter to match the normalized Hamming weight of the overall Bloom filter.

Figure 5.16 shows that the normalized Hamming weights over all ECC words selected this way is much closer to the desired distribution shown in Figure 5.10. This implies that less queries are identified as different and thus a larger number is

classified as a possible inference attack. This might also lead to a larger number of false-positives, but also a better actual matching of close queries to detect ongoing inference attack queries. As a result, the complete database all-to-all comparison of fuzzy commitments using the uniform mapping algorithm revealed 970 identically decoded strings, which is still a very low false-positive attack detection rate of 0.000013 and did therefore not significantly change the results compared to the consecutive (or sequential) ECC selection over the Bloom filter.

Before coming to the third permutation algorithm or mapping technique from Bloom filters to ECC words, we will describe and analyze possible attacks on a privacy-preserving comparison scheme, a method to simulate such an attack and how the simulation is used to determine the attack detection capability of our inference attack detection algorithm.

### 5.8.1. Matching attacks

Up to now we analyzed how well genuine queries were preserved without raising false alarms — as represented by the entries in the DNA database. However, it is also necessary to give performance results on how well ongoing attacks are matched. Therefore a description of an attack is needed. An attacker that performs an inference attack is very likely to query rather similar queries, with a low distance between them. The distance can either be measured as Hamming distance over the submitted Bloom filter, or as the edit distance over the original genomic sequence or character string. In the end, we want queries that are too close to previously issued queries (having a minimum pair-wise edit distance below $d_{\mathrm{E}_{\mathrm{low}}}$) to be detected and rejected with high probability, while queries with a rather high distance to all previously issued requests to go through undetected (minimum pair-wise edit distance above $d_{\mathrm{E}_{\mathrm{high}}}$), also with high probability.

To simulate such an attacker, the sampled distance results from the protocol in Chapter 4, as they are visualized in Figure 4.4 are fitted to a Gaussian function for each edit distance value in the range $[1, 100]$. Let $G_i$ denote the fitted Gaussian for edit distance $i \in 1, \ldots, 100$. The actual simulated attack then works like this: select a random element from the database, translate it into a Bloom filter $b$ and perform an edit operation (with edit distance $d_{\mathrm{E}}$) on it by sampling from the fitted Gaussian $G_{d_{\mathrm{E}}}$ and flipping the returned number of bits at random positions $b$ to create the attack query $b'$.

Both queries $b, b'$ are mapped to a fuzzy commitment via error-correction and decoding using the Error-Correcting Code. For small values $d_{\mathrm{E}} \leq d_{\mathrm{E}_{\mathrm{low}}}$ the queries should be mapped to the same commitment, or equivalently to the same information words after decoding. On the opposite, larger distances $d_{\mathrm{E}} \geq d_{\mathrm{E}_{\mathrm{high}}}$ should produce a mapping to different information words and thus commitments.

Evaluating this attack over some database entries ($> 100$), different edit distances and involving several repetitions ($> 100$), produced an unsatisfying attack detection performance. As shown above, nearly all genuine queries passed, but also very close queries were not detected satisfyingly. Even requests with just a single bit change in the Bloom filter were detected as attacks for only half of the queries. This behavior of not detecting close queries with high probability and thus the inability of detecting attacks based on the query similarity is not suited to detect inference attacks in our scenario. As even very close requests passed the system 50% of the time the root cause of the detection problems must be found.

The reason for the bad performance lies in the decoding of random bit strings for the ECCs. Even though the distribution of normalized Hamming weights was balanced using sampled indices from the Bloom filter to map bits to an ECC — as depicted in Figure 5.16, the to-be-decoded bit strings are still random. Recall that the used Reed-Muller code with a configuration of $m = 10$ and thus a $d_{\mathrm{H}_{\min}} = 2^{m-1} = 512$ can correct up to $t = \lfloor \frac{d_{\mathrm{H}_{\min}}-1}{2} \rfloor = 2^{m-2} - 1 = 255$ bit errors upon a codeword of length $2^m = 1024$ bits. Each codeword has a Hamming weight of $2^{m-1}$ and thus a uniformly random bit string of length $2^m$ will have on average a Hamming distance of $2^{m-2}$ from each codeword. Except for the two codewords, which are composed of only 0's and only 1's, here the Hamming distance is $2^{m-1}$.

However, if the random bit string has a Hamming distance of $2^{m-2}$ from any codeword, and the code can correct up to $2^{m-2} - 1$ errors, then the ECC words reside just on the edge to decode into different information words. In case only a few bits are changed upon these pseudo-random bit strings a decoding into a different information word is highly probable. This is not the desired behavior, as decoding should be identical for two bit string with a low Hamming distance.

### 5.8.2. Close to Codeword Mapping

So, the issue of the bad detection performance lies in the random distribution of ECC word bits, resulting in a bit string which lies between codewords (geometrically speaking). The logical consequence is to derive ECC words, which are closer to a codeword than random strings and thus can tolerate more bit flips before decoding into different information words.

**3. Close to codeword** To correct this phenomena, we no longer sample indices for the ECCs words randomly or choose them to fit a desired distribution of normalized weights, but instead select indices to get a bit string with a certain Hamming distance to a pre-selected codeword. This way we are sure that the ECCs can

decode also bit strings within a certain Hamming distance to the strings selected by the indices.

The index selection works like this:

- Select the number $r$ of ECC words that are used to represent and decode the Bloom filter $b$. ($r = 22$ in our case.)
- Generate $r$ random information words $x_i = \{0,1\}^{m+1} : \ 1 \leq i \leq r$.
- Encode $x_i$ to codewords $y_i$.
- Select a vector $\text{idx}_i$ containing $2^m$ indices from $b$ for each codeword $y_i$ such that the concatenation of all bits $b_{\text{idx}_i}$ selected by the index vector $\text{idx}_i$ equals the codeword $y_i$.

Depending on the actually chosen Error-Correcting Code, some special cases must be avoided. One of them is that for a Reed-Muller ECC, there exist the codewords $\{1\}^m$ (a bit sequence of 1's with length $m$ bits) and $\{0\}^m$, which should be avoided. Due to drawing many ones or zeroes from the Bloom filter $b$ there are possibly not enough values left to select the indices for the remaining ECC words. In any case, index selection might run out of 1's or 0's in the Bloom filter, in which case the selection must be run again on new "random information words" $x_i$. To reduce the chance of running out of bit values, the number of ECC words was reduced from $r = 22$ down to $r = 20$, giving a reduced length of $l_r = 2^m \cdot r = 20480$ bits. This still allows to differentiate $2^{220}$ ($2^{110}$ respecting the birthday paradox) input sequences, which should be enough in practice.

The reference Bloom filter $b$ used to select the $k$ vectors of indices is constructed from the rCRS reference sequence. As mentioned before, a reference sequence is used to not leak any information about the input of the client or the server and to be able to compare results. It follows directly from the used ECC and its configuration (first-order Reed-Muller with $m = 10$) and the mapping of the reference Bloom filter to codewords, that other Bloom filters with a Hamming distance of up $2^{m-2} - 1 = 255$ bits from the reference Bloom filter are decoded identically and always generate the same fuzzy commitment. Furthermore, as not all bits of the Bloom filter are used and decoded using an ECC, a change in these $l - l_r = 2356$ unused bits will not be detected.

Recalling the Distribution of Hamming distances over all Bloom filters generated from the database, plotted in Figure 5.12, showed that a majority of the occurring distances for genuine sequences can be found in the range $[400, 1000]$ for a Bloom filter length of 22836 bits. Each ECC therefore receives on average a Hamming distance between 17.9 and 44.8 bits between two correct queries, which should be decoded into different information words each with high probability. It would be enough if one of the ECCs would decode into a different information word with high probability for two genuine (non-inference attack) requests.

However, as each of the $r$ ECCs corrects $2^{m-2}-1 = 255$ flipped bits, all genuine requests will be mapped to the same information word. This removes the utility of the scheme as all genuine queries are detected as identical and thus cannot be differentiated from smaller distances. Luckily this behavior can be nicely balanced and adopted to the error-correction and therefore similarity detection requirements defined through the underlying data. In the setup process of the inference control scheme, a error-correction capability of the scheme must be fixed, which influences the final detection capability and the accuracy of detecting attacks, as well as the false-positive rate in doing so. If each ECC instance should be able to group strings with a Hamming distance of $o$ around the reference bit string generated by the rCRS, its error-correcting capabilities must be restricted according to $o$. This can easily be done by a simple adaptation of the aforementioned steps.

Instead of selecting the indices so that the resulting bit mapping to ECC words resembles codewords exactly, only $2^m - \left(\left(2^{m-2}-1\right) - o\right)$ bits are chosen to exactly match the Bloom filter. The remaining $\left(2^{m-2}-1\right) - o$ bits are chosen deliberately wrong. By this construction, ECC word bit string that differ in $o' > o$ bits from the reference are not guaranteed to be corrected to the same information word anymore. The higher the difference $o' - o$ with $o' > o$, the higher is the chance to fall into another information word after decoding.

From the distribution of Hamming distances over all possible Bloom filter generated out of the database (Figure 5.12), we chose $o = 20$ for testing and generated information words over the database, as well as ran the described attack simulation by sampling the Gaussian functions and perform decoding comparisons (just as described above). The actual value $o = 20$ was chosen as this allows for a total Hamming distance of up to 400 bits to be changed over the whole Bloom filter without changing the commitment. This roughly equals an edit distance of up to 19 to be mapped to the same commitment, in theory.

Generating all commitments for the whole database and checking them pair-wise yields a false-positive rate of 0.017 to detect genuine queries as inference attack. Figure 5.17 puts the distributions of pair-wise Hamming weight distances into relation. The distributions were generated using *sequential* mapping of Bloom filter bits to each ECC, a *uniform* random index selection to derive Bloom filter to ECC mappings and close to *codeword* mappings. The Hamming distance between information words over the database varies depending on the actually used mapping scheme.

Following these results, the above described inference attack scenario is run. Figure 5.18 presents the results obtained from comparing information words of many similar queries and different values $o \in \{10, 15, 20\}$. Looking at $o = 20$ for example, the scheme is able to detect close strings with an edit distance of 1 with a probability of 0.7498, however strings with an edit distance of about 20 — which

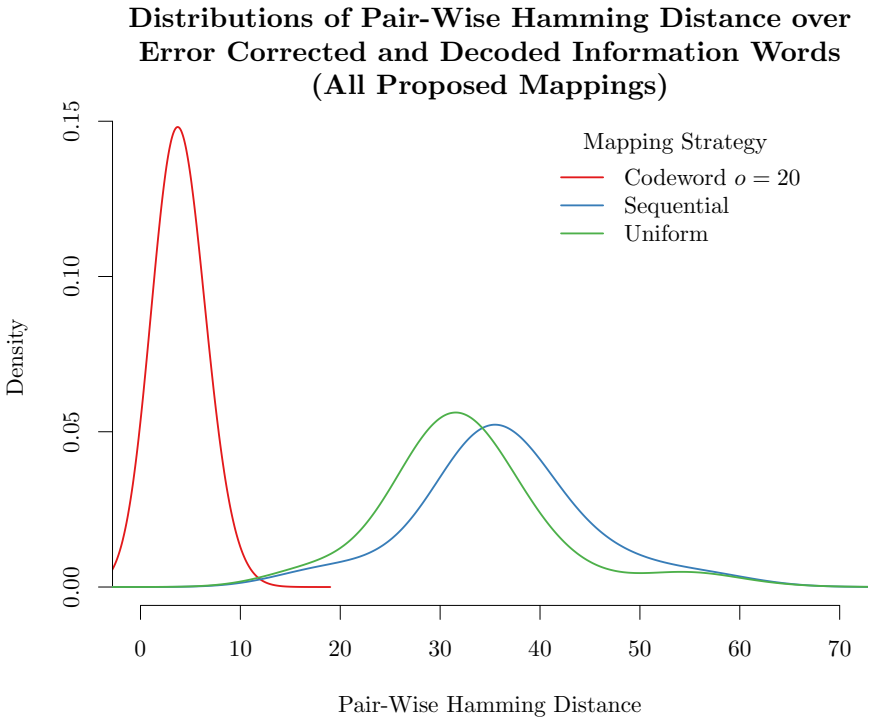



Figure 5.17.: Distribution of Hamming distances between all decoded Bloom filters generated from the "Human Mitochondiral genome Database" [LWY07].

should probably not count as an inference attack — are only detected as similar with a probability of 0.0143. The steep curve indicates a quick transition from the range of distances that exhibit a high probability of being detected as similar over to the range of distances, which is unlikely to be identified as similar. Such a behavior is highly desired, as it allows a separation between attacks and genuine queries.

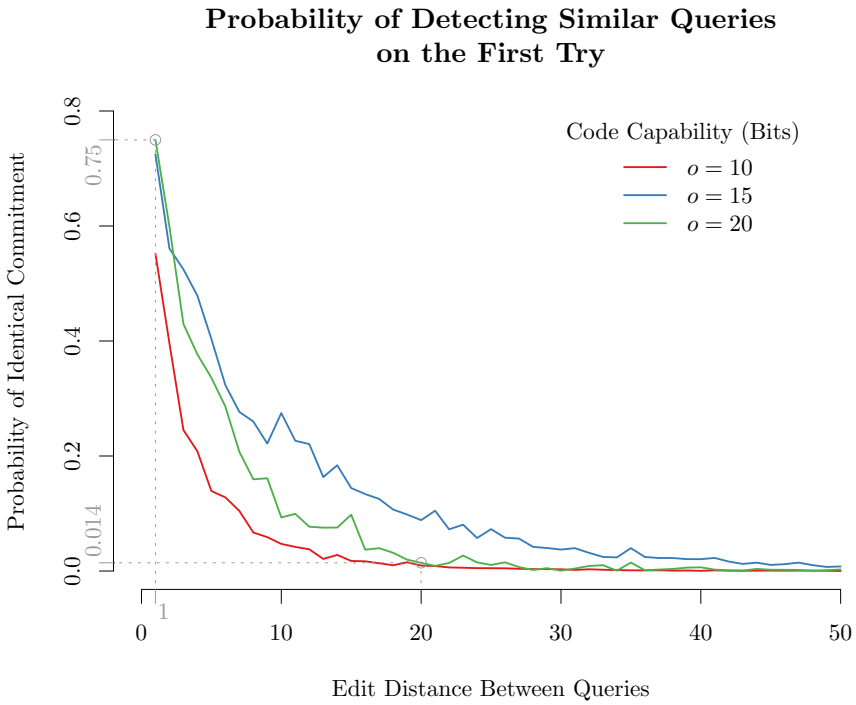



Figure 5.18.: Probability of mapping two close queries to the same commitment. Closeness is defined as edit distance between the underlying strings. The edit distance to Hamming distance mappings are sampled from fitted Gaussians as described above.

## 5.9. Conclusion

We investigated how to mitigate inference attacks on privacy-preserving genome matching. Since randomization approaches, such as differential privacy, cannot work, we employ a detection technique using an ECC and a one-way function. A novel zero-knowledge proof ensures honesty on part of the querier. This zero-knowledge proof is more efficient than generic techniques by using somewhat homomorphic encryption and also more efficient than directly proofing the error-correction by introducing the codeword.

We show an analysis that our framework (in combination with limiting the number of queries) is able to differentiate between similar and non-similar queries regarding the Hamming distance. As such it is able to detect and mitigate inference attacks that use similar queries to quickly derive a solution. The scheme can be adopted easily to be more restrictive against inference attacks by increasing the offset parameter $o$ or by lowering the normalized Bloom filter weight via increasing

the Bloom filter length. Furthermore, the false-positive detection rate could be reduced — to allow highly similar queries to go through — by choosing a smaller $m$ for the ECC and thus a possibly larger input domain, at the price of having a lower accuracy for detecting queries for inference attacks. A strong measure to influence the detection performance is the possibility to describe a custom selection of bit string indices for mapping towards the ECCs.

The scheme works best when similar elements, regardless if these might be strings or something else, are of rather similar size, or have a similarly weighted feature vector, which can be used as input instead of the Bloom filter. Future work might include the adoption of ECCs of different sizes to utilize nearly all Bloom filter bits and thus reduce sampling distortion. Furthermore, it is of interest how far different algorithms from coding theory and parameter variations influence the detection capability for similar elements. Applying the presented techniques to other domains apart genome or even string matching from other domains would be interesting.

# 6. Homomorphic Chipertext Compression

Parts of this chapter are published in [Bec15].

## 6.1. Introduction

As reasoned during the requirement discussion in Section 1.2, a client using the comparison scheme described in Chapter 4 and 5 might be resource-constrained or connected via a low bandwidth (wireless) channel. In either case it is necessary to waste as few resources as possible for encryption or transmission.

Efficient encryption, storage and communication while still being able to use secure operations (*e.g.* via a Homomorphic Encryption (HE) systems) is needed. What comes to mind is a combination of symmetric and asymmetric cryptography — similar to the construction of a hybrid cryptographic system as often used by integration of [Cal+07] and [Kal98]. In the hybrid cryptographic system case, an asymmetric system is used to encrypt a symmetric key $k_s$, which is again used for a symmetric cryptographic system to encrypt the actual data. Through such a construction, properties from the asymmetric system are combined with the efficiency of symmetric cryptographic.

Combining a symmetric cryptosystem with an HE system for increased transmission and storage efficiency was mentioned in several papers, as already discussed in Section 2.4. Data is encrypted using a symmetric transmission-cipher before sending it to a remote server for further processing under a compute-cipher. A transition function transforms data encrypted under the transmission-cipher into data encrypted under the compute-cipher. The relevant literature uses semantically secure symmetric encryption schemes as transmission ciphers and evaluate their respective decryption functions homomorphically for transmission- to compute-cipher transition.

Such combinations are either proposed to increase transmission efficiency directly [BGV11], or to compare HE schemes regarding their performance [GHS12b; DHS14; LCT14; Che+13].

We describe a much more efficient construction in which a stream cipher is used to generate a pseudo-random keystream that encrypts the plaintext using a single operation. As a result, our transition function has a much improved runtime, which is several orders of magnitude lower than for the related work, while achieving similar goals for encryption, transmission and storage efficiency. The huge efficiency gain comes from the minimal decryption circuit, a single homomorphic operation, required for homomorphic transition from the transmission-cipher to the compute-cipher. Homomorphic evaluation of this single-operation circuit is of course much faster than evaluating a decryption circuit of a symmetric block cipher (*e.g.* Advanced Encryption Standard (AES)).

The next Section 6.2 describes the prerequisites needed further on in the construction of the scheme, which is then given in Section 6.3. Section 6.4 describes a security proof by reducing the assumptions to the ones used by the underlying Pseudo-Random Bit Generator (PRBG) and HE system, while Section 6.5 presents performance results and comparisons to previous proposals.

## 6.2. Preliminaries

Several pseudo-random generators will be used throughout the next sections. These are separated by their use or co-domain in which they pseudo-randomly generate values or elements. A short list will introduce the different generators for disambiguation. The exact definition including the domain and co-domain is given later.

PRBG Pseudo-Random Bit Generator. A function generating a stream of pseudo-random bits (see Section 6.2.1).

PRNG Pseudo-Random Number Generator. A function generating a stream of pseudo-random real values (in this scheme the interval $[0, 1]$ is used) (see Definition 6.4.1).

PREG Pseudo-Random Element Generator. A function generating a stream of pseudo-random elements from a specific group — used to generate random ciphertext elements (see Definition 6.4.1).

PRKG Pseudo-Random Key-Stream Generator. A function generating a stream of pseudo-random values used as key-stream for symmetric encryption in the presented scheme (see Section 6.3.1).

### 6.2.1. Pseudo-Random Bit Generator (PRBG)

Let $g\colon \{0,1\}^l \to \{0,1\}^m$ be a cyptographically strong PRBG that produces pseudo-random bit values, which cannot be differentiated from a independent and identically distributed (i.i.d.) uniform random bit stream by a deterministic polynomial time (PTIME) restricted attacker. Let $x = g(s_0)$ with $s_0 \in \{0,1\}^l$ being the seed with length $l$ bits and $x$ being the pseudo-random bit sequence of length $m$. As most such algorithms are designed to efficiently run on processors with a certain word size, they most often use, process and output pseudo-random bit sequences at the word size (or a multiple of it) of the underlying processor. Let $f\colon \{0,1\}^l \times \mathbb{Z} \to \mathbb{Z}_{2^w}$ be a function defined by $f(s_0, i) = g(s_0)_{[i\cdot w:(i+1)w-1]}$, which returns $w$ bits for each $i$. The $w$ bits can be mapped via a bijection to an element in $\mathbb{Z}_{2^w}$.

### 6.2.2. Homomorphic Encryption (HE)

The notation introduced in 3.5 is used, with a brief recall of the important aspects. A HE scheme is a quadruple $\mathcal{H} = (\mathsf{KeyGen}_\mathcal{H}, \mathsf{Enc}_\mathcal{H}, \mathsf{Dec}_\mathcal{H}, \mathsf{Eval}_\mathcal{H})$ of PTIME algorithms. $\mathsf{KeyGen}_\mathcal{H}\colon \mathbb{Z} \to K_S \times K_P$ is the key generation function, which takes a security parameter $\kappa \in \mathbb{Z}$ and outputs the secret key $\mathsf{sk} \in K_S$ and public key $\mathsf{pk} \in K_P$. $\mathsf{Enc}_\mathcal{H}\colon K_P \times P_\mathcal{H} \to C_\mathcal{H}$ describes the probabilistic encryption function with $\mathsf{Enc}_\mathcal{H}(\mathsf{pk}, m) = c$ and equivalently $\mathsf{Dec}_\mathcal{H}\colon K_S \times C_\mathcal{H} \to P_\mathcal{H}$ the decryption function with $\mathsf{Dec}_\mathcal{H}(\mathsf{sk}, c) = m'$, $m, m' \in P_\mathcal{H}$ being plaintexts, $c \in C_\mathcal{H}$ being a ciphertext. $\mathsf{Eval}_\mathcal{H}\colon K_P \times \mathcal{F} \times C^*_\mathcal{H} \times P^*_\mathcal{H} \to C_\mathcal{H}$ describes a function that performs homomorphic operations upon the given ciphertexts to evaluate a certain functionality. It is given by $\mathsf{Eval}_\mathcal{H}(\mathsf{pk}, F, \bar{c}, \bar{p})$ for a function $F \in \mathcal{F}$ given some ciphertexts $\bar{c} \in C^*_\mathcal{H}$, and some plaintexts $\bar{p} \in P^*_\mathcal{H}$.

It is sufficient to use a Partly Homomorphic Encryption (PHE) system for the purposes in this chapter. In particular we will base the following descriptions on a Additive Homomorphic Encryption (AHE) scheme, but Multiplicative Homomorphic Encryption (MHE) schemes are equally usable. Further details on the actual choice of HE system will follow in Section 6.5.

## 6.3. Design

We describe the construction of our stream cipher after recalling the use case. A client wants to outsource computation to a server and uses a HE system to hide the sensitive data from the less trusted compute server. He chooses a cryptographic

system that can evaluate the desired functionality homomorphically, however instead of encrypting the sensitive data using the HE system directly, a corresponding stream cipher is constructed as described in this section and used instead to encrypt all the sensitive data. All necessary (public) plaintext data together with the symmetrically encrypted private input is send to the compute server. The server translates the symmetrically encrypted data into homomorphically encrypted data and performs the actual calculation. All results are transferred back to the client for decryption. As the stream cipher depends on the actual HE scheme $\mathcal{H}$, the definition of such a scheme is taken from Section 3.5.

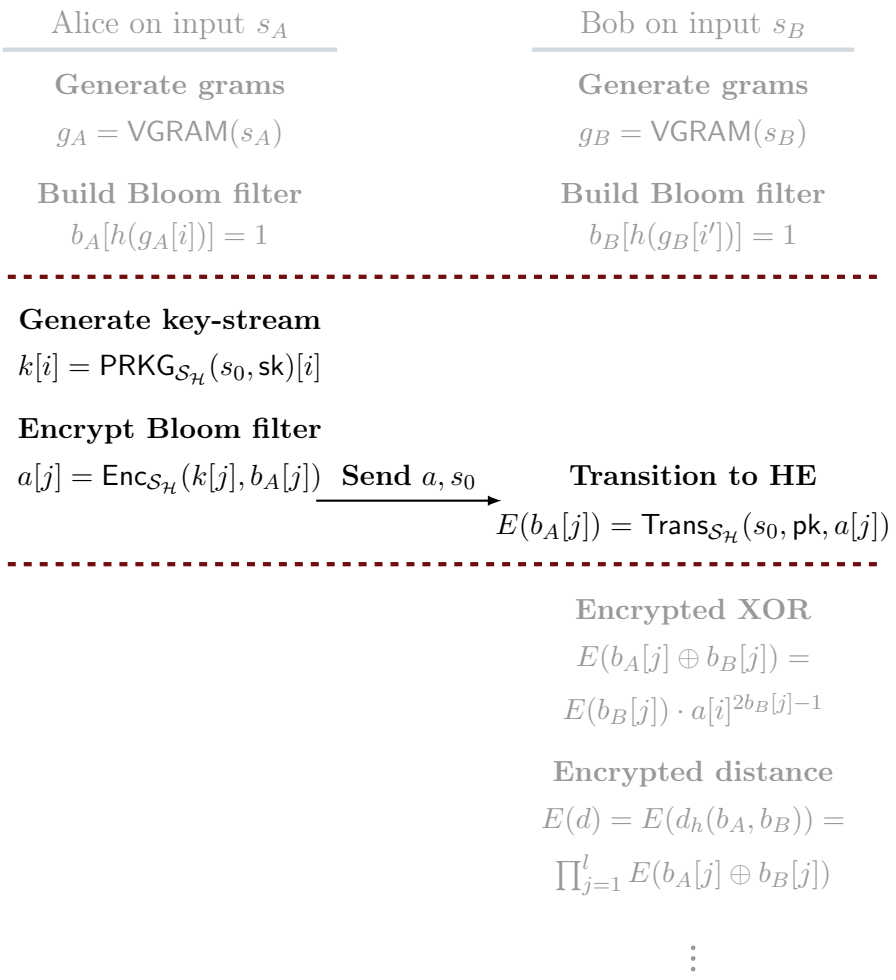



Figure 6.1.: Extension for privacy-preserving distance measure protocol presented in Chapter 4. The client generates a key-stream, which is used as a pseudo one-time pad. At the server side, this key-stream can be efficiently generated within an HE scheme to transform the symmetrically encrypted data into homomorphically encrypted data.

Figure 6.1 depicts the described protocol in between the red dashed lines and embeds it in the string matching scheme designed in Chapter 4.

We define the stream cipher, which is simply a modular subtraction or addition of a pseudo-random key-stream with the actual data. The contained Pseudo-Random Element Generator (PREG) is the most important part of the construction, but will be skipped for now and defined later for ease of presentation. Likewise the definition of the transition function, based upon the PREG, will be given later.

**Definition 6.1: Randomized Homomorphic Stream Cipher**

A stream cipher $\mathcal{S}_\mathcal{H}$ for a Homomorphic Encryption (HE) system $\mathcal{H}$ is a quadruple of PTIME algorithms $\mathcal{S}_\mathcal{H} = (\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}, \mathsf{Enc}_{\mathcal{S}_\mathcal{H}}, \mathsf{Dec}_{\mathcal{S}_\mathcal{H}}, \mathsf{Trans}_{\mathcal{S}_\mathcal{H}})$. $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}: Z_{2^n} \times K_S \to P^*_\mathcal{H}$ is the Pseudo-Random Key-Stream Generator (PRKG), which generates the pseudo-random key-stream used for encryption. $\mathsf{Enc}_{\mathcal{S}_\mathcal{H}}: P_\mathcal{H} \times P_\mathcal{H} \to P_\mathcal{H}$ is used to encrypt data with $\mathsf{Enc}_{\mathcal{S}_\mathcal{H}}(u_{P_\mathcal{H}}, m) = d$ and $u_{P_\mathcal{H}} \in P_\mathcal{H}$ being a uniform random variable. Likewise $\mathsf{Dec}_{\mathcal{S}_\mathcal{H}}: P_\mathcal{H} \times P_\mathcal{H} \to P_\mathcal{H}$ is used to decrypt the ciphertext $\mathsf{Dec}_{\mathcal{S}_\mathcal{H}}(u_{P_\mathcal{H}}, d) = m'$. Furthermore, $\mathsf{Dec}_{\mathcal{S}_\mathcal{H}}(u_{P_\mathcal{H}}, \mathsf{Enc}_{\mathcal{S}_\mathcal{H}}(u_{P_\mathcal{H}}, m)) = m$ must be fulfilled for correct decryption with $u_{P_\mathcal{H}}$ being the same (pseudo-) random value for encryption and decryption. The function $\mathsf{Trans}_{\mathcal{S}_\mathcal{H}}: Z_{2^n} \times K_P \times P_\mathcal{H} \to C_\mathcal{H}$ translates a value $m$ encrypted under the described stream cipher $\mathcal{S}_\mathcal{H}$ with $Enc_{\mathcal{S}_\mathcal{H}}(u_{P_\mathcal{H}}, m)$ into an encryption under the HE system $\mathcal{H}$ as $\mathsf{Enc}_\mathcal{H}(\mathsf{pk}, m)$.

As $P_\mathcal{H}$ is countable for cryptographic systems, all elements $p_i \in P_\mathcal{H}$ for $1 \leq i \leq |P_\mathcal{H}|$ can be indexed. This way a modular addition for $p_i, p_j \in P_\mathcal{H}$ over $P_\mathcal{H}$ is defined as $p_i + p_j = p_{i+j \mod |P_\mathcal{H}|}$. Modular subtraction is defined equivalently. Using addition and subtraction we define $\mathsf{Enc}_{\mathcal{S}_\mathcal{H}}(p_i, p_j) = p_j - p_i$ and $\mathsf{Dec}_{\mathcal{S}_\mathcal{H}}(p_i, p_j) = p_j + p_i$.

### 6.3.1. Pseudo-Random Key-Stream Generator (PRKG)

We construct a function $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}$, that generates pseudo-random ciphertexts $u_{C_\mathcal{H}} \in C_\mathcal{H}$ using a Pseudo-Random Element Generator (PREG), decrypts the pseudo-random elements and returns the resulting plaintexts ($\mathsf{Dec}_\mathcal{H}(\mathsf{sk}, u_{C_\mathcal{H}}) \in P_\mathcal{H}$) with the goal of using the plaintext values as a key-stream for the stream cipher. Therefore a PREG to generate random ciphertexts is necessary. Let $\mathsf{Gen}_{C_\mathcal{H}}: \{0,1\}^* \to C^*_\mathcal{H}$ be a PTIME function generating ciphertexts uniform in $C_\mathcal{H}$ given a seed $s_0 \in \{0,1\}^*$ as input. Furthermore, let $\mathsf{Gen}_{C_\mathcal{H}}(s_0)[i]$ and similar $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}(s_0, \mathsf{sk})[i]$ denote the $i$th element generated with $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}(s_0, \mathsf{sk})[i] = \mathsf{Dec}_\mathcal{H}(\mathsf{sk}, \mathsf{Gen}_{C_\mathcal{H}}(s_0)[i])$.

Encryption of plaintext values $m_j \in P_{\mathcal{H}}$ under the stream cipher $\mathcal{S}_{\mathcal{H}}$ is defined as $d_j = \mathsf{Enc}_{\mathcal{S}_{\mathcal{H}}}(\mathsf{PRKG}_{\mathcal{S}_{\mathcal{H}}}(s_0, \mathsf{sk})[j], m_j)$, using the public seed $s_0 \in \{0,1\}^*$ for the PRKG and the secret key $\mathsf{sk}$ of the HE system $\mathcal{H}$.

The transition function $\mathsf{Trans}_{\mathcal{S}_{\mathcal{H}}}$ (translating a symmetrically encrypted value $d_j \in P_{\mathcal{H}}$ into an homomorphically encrypted one $c_j \in C_{\mathcal{H}}$) is then defined as

$$c_j = \mathsf{Trans}_{\mathcal{S}_{\mathcal{H}}}(s_0, \mathsf{pk}, d_j) \tag{6.1}$$

$$= \mathsf{Eval}_{\mathcal{H}}(\mathsf{pk}, \oplus, (\mathsf{Enc}_{\mathcal{H}}(\mathsf{pk}, d_j), \mathsf{Gen}_{C_{\mathcal{H}}}(s_0)[j]))). \tag{6.2}$$

As such $\mathsf{Trans}_{\mathcal{S}_{\mathcal{H}}}$ takes the public seed $s_0$ also used in the symmetric encryption algorithm $\mathsf{Enc}_{\mathcal{S}_{\mathcal{H}}}$, the public key $\mathsf{pk} \in K_P$ of the HE system $\mathcal{H}$ and the symmetrically encrypted value $d_j$ to construct the asymmetric encryption of the underlying plaintext value $m_j$. $\oplus$ defines the function that evaluates addition over $P_{\mathcal{H}}$ and thus resembles the decryption function of the cryptosystem $\mathcal{S}_{\mathcal{H}}$.

We introduced in this chapter a private-key encryption scheme that is tightly bound to a chosen homomorphic encryption system, has no ciphertext expansion and the property of allowing efficient transition for an untrusted party from symmatrically encrypted data into an encryption under the homomorphic cryptosystem. Decryption can therefore be performed in two different ways. Either by conversion to the inherent homomorphic encryption scheme and its decryption function, or similarly to the actual encryption through the evaluation of the PRKG without the use of the transition function.

### 6.3.2. Encryption Efficiency

For the client to encrypt data using the stream cipher, it evaluates a PREG $\mathsf{Gen}_{C_{\mathcal{H}}}$, the decryption function $\mathsf{Dec}_{\mathcal{H}}$ and the encryption function of the stream cipher $\mathsf{Enc}_{\mathcal{S}_{\mathcal{H}}}$. The computationally largest part of the encryption is the evaluation of the decryption function $\mathsf{Dec}_{\mathcal{H}}$. Depending on the cryptosystem $\mathcal{H}$, computing $\mathsf{Enc}_{\mathcal{H}}$ may be easier than computing $\mathsf{Dec}_{\mathcal{H}}$. However, in our setting everything except $\mathsf{Enc}_{\mathcal{S}_{\mathcal{H}}}$ can be pre-computed by the client. That is, the expensive functions $\mathsf{Dec}_{\mathcal{H}}$ and to a lesser extend also $\mathsf{Gen}_{C_{\mathcal{H}}}$ can be evaluated while there is no power constraint for the client (*e.g.* during charging the battery), or by a trusted more powerful system from the same party (*e.g.* a laptop that the device is synced with occasionally). The client then only stores the key-stream generated by $\mathsf{PRKG}_{\mathcal{S}_{\mathcal{H}}}$ and the seed $s_0$.

This way a power- and resource-constrained client must not evaluate any function from $\mathcal{H}$ during encryption. Devices previously incapable of producing homomorphic encryptions due to resource constraints can be used to outsource secure computation. Pre-computation of the key-stream will further greatly reduce encryption latency.

### 6.3.3. Key-Stream Pre-Computation

Having a threshold homomorphic encryption scheme allows to even outsource the pre-comuptation of the key-stream to $n$ untrusted parties and is secure against collusion as long as only a minority, that is up to $\lfloor (n-1)/2 \rfloor$ parties are colluding [CDN01]. In a threshold HE scheme the secret key can be split into $n$ parts to have a distributed decryption of a ciphertext. Thus the pre-computation of a key-stream is given to $n$ parties and the results are combined at the client side for further usage.

## 6.4. Security Analysis

To analyze the security of the proposed cryptographic system, we consider relevant parts of the construction, prove their security and combine obtained results.

The encryption function $\mathsf{Enc}_{S_\mathcal{H}}\colon P_\mathcal{H} \times P_\mathcal{H} \to P_\mathcal{H}$ of the stream cipher $S_\mathcal{H}$ is a vernam cipher, which means that if the used key-stream consists of elements chosen uniform, independent and randomly from $P_\mathcal{H}$, the encryption is information-theoretically secure. However, as the key-stream is generated pseudo-randomly, we have to analyze the generator. According to Section 6.3, key-stream elements are decryptions of elements from $C_\mathcal{H}$ and a computationally restricted adversary must not be able to tell them apart from independently chosen uniform random elements from $P_\mathcal{H}$ using any algorithm with runtime polynomially bound in the size of the seed $s_0$ of the generator.

As $\mathsf{Enc}_{S_\mathcal{H}}$ must use uniformly distributed key-stream values, it is necessary and sufficient for $\mathsf{Dec}_\mathcal{H}\colon K_S \times C_\mathcal{H} \to P_\mathcal{H}$ to return uniformly distributed elements in $P_\mathcal{H}$. Let $\gamma$ denote the necessary input distribution over $C_\mathcal{H}$ for $\mathsf{Dec}_\mathcal{H}$, and $\pi$ the output distribution over $P_\mathcal{H}$ given $\gamma$ as input. $\pi$ should be computationally indistinguishable from the uniform distribution $\upsilon$. Furthermore, the asymmetric decryption key $\mathsf{sk} \in K_S$ must be independent from $\gamma$ and indistiguishability between $\pi$ and $\upsilon$ must hold for all possible secret keys $\mathsf{sk}$.

However, it might not always be trivial, even if the distribution $\gamma$ together with the domain of $\mathsf{Dec}_\mathcal{H}$ is known, to generate $\gamma$-distributed values within $C_\mathcal{H}$. In Section 6.3 a cryptographically secure PREG is used to generate uniform input for $\mathsf{Dec}_\mathcal{H}$. This is fine as long as $\gamma$ is computationally indistinguishable from the uniform distribution, which is true for many partially homomorphic encryption schemes like [Pai99; NS98; RSA78; ElG84; GM82; Ben94; DJ01; BGN05; OU98], which are group-homomorphic as given by Armknecht, Katzenbeisser, and Peter

[AKP13], or more specifically, that use modular exponentiation over a residue class. A simplified security proof is given at the end of this section.

For homomorphic cryptosystems with $\gamma$ not being the uniform distribution, one way of generating $\gamma$-distributed random values, using a cryptographically secure Pseudo-Random Number Generator (PRNG) that outputs uniformly distributed values, is to use inverse transform sampling [Dev86]. This means that one must be able to construct an inverse distribution function (also called inverse cumulative distribution function, quantile function, or percent point function) $G_{\mathcal{H}}: [0,1] \to C_{\mathcal{H}}$. Let a discrete random variable $x$ replace the homomorphic encryption function $\mathsf{Enc}_{\mathcal{H}}$. Given a uniform input $x \in \mathbb{R}$ with $0 \leq x \leq 1$, $\mathsf{G}_{\mathcal{H}}(x)$ produces $\gamma$-distributed elements in $C_{\mathcal{H}}$.

If the Cumulative Distribution Function (CDF) $\mathsf{F}_{\mathcal{H}}: C_{\mathcal{H}} \to [0,1]$ for $\mathsf{Enc}_{\mathcal{H}}$ has a closed form, the inverse $G_{\mathcal{H}} = \mathsf{F}_{\mathcal{H}}^{-1}$ can be found using root-finding algorithms [Dev86; Pre+07]. If there exists no closed form inverse, approximations using numerical methods [Pre+07] satisfying Assumption 6.4.1 can be used.

### 6.4.1. Arbitrarily Distributed Random Ciphertexts

We will prove the security of our proposed stream cipher in a general setting using Assumption 6.4.1. This is done by showing that breaking the stream cipher encryption implies breaking either the Homomorphic Encryption (HE) scheme, or the Pseudo-Random Bit Generator (PRBG).

First we introduce definitions for the security of pseudo-random generators and indistinguishability upon their output. A PTIME algorithm $A: S \to \{0,1\}$ is used to describe the advantage $\mathrm{Adv}\, A = |\mathsf{P}[A(x) = 1] - \mathsf{P}[A(y) = 1]|$ with $x, y \in S$ that exists in distinguishing two inputs $x$ and $y$.

**Definition 6.2: Computationally Secure PRNG**

A PRNG $g: \{0,1\}^m \to K$ is called computationally secure if $\mathrm{Adv}\, A \leq \epsilon$ to distinguish $x = g(u_S)$ from $y = u_K$ for uniform random variables $u_S \in \{0,1\}^m, u_K \in K$ for any PTIME algorithm $A$.

For the rest of the chapter algorithms are assumed to run in deterministic polynomial time (PTIME), even if it is not made explicit. Following Definition 6.4.1, there exists no PTIME algorithm that can distinguish the output of a *computationally secure* Pseudo-Random Number Generator (PRNG) $g$ from uniform random values with probability higher than $\epsilon$. Observe that the co-domain $K$ of $g$ used in Definition 6.4.1 may be a sequence of elements from another set, *e.g.* $\mathbb{R}^{\hat{m}}$. The

length of this sequence is polynomially bounded in the logarithm of the cardinality of the domain of $g$ (the seed in $S$) [GB08]. The size of each element is equivalently polynomially bounded.

To include HE systems that have no closed form inverse of their $\mathsf{Enc}_{\mathcal{H}}$ CDF, we introduce an assumption about an indistinguishable approximation. Whether this assumption is too strong in general is subject to further research, however it allows us to include certain somewhat, leveled and fully homomorphic asymmetric cryptographic systems in the security proof.

**Assumption 6.1: Approximate Quantile Function**

It exists a PTIME function $G_{\approx}\colon [0,1] \to C_{\mathcal{H}}$ with a uniform random variable $u_{\mathbb{R}} \in [0,1]$ as input, which approximates the quantile function $G\colon [0,1] \to C_{\mathcal{H}}$ with input being also a uniform random variable $u'_{\mathbb{R}} \in [0,1]$. $G_{\approx}$ is called computationally indistinguishable by a polynomially-bounded adversary $A$ from the result of $G$ given a uniform i.i.d. random variable as input if the probability of distinguishing the output of $G(u'_{\mathbb{R}})$ from $G_{\approx}(u_{\mathbb{R}})$ is bounded by $\epsilon$.

This assumption is easily fulfilled for encryption functions, which generate uniformly distributed ciphertexts when encrypting a uniform random variable. In such cases, the quantile function $G$ can directly be constructed and evaluated in PTIME and as such $G_{\approx}$ would be identical (and thus indistinguishable) to $G$. In fact, the use of $G$ or $G_{\approx}$ together with a generator of elements in $\mathbb{R}$ in the interval $[0,1]$ can also be simplified by using a secure Pseudo-Random Element Generator (PREG) to directly generate elements in the ciphertext space $C_{\mathcal{H}}$

The definition of a computationally secure PREG is similar to, and follows the definition of the computationally secure PRNG. It just differs in the co-domain and output distribution.

**Definition 6.3: Computationally Secure PREG**

$\mathsf{Gen}_{C_{\mathcal{H}}}\colon \{0,1\}^k \to C_{\mathcal{H}}^l$ is a computationally secure PREG, if Adv $A \leq \epsilon$ to distinguish $x = Gen(r_{\{0,1\}^k})$ from $y = r_{C_{\mathcal{H}}}$ for uniform, random and independent values $r_{\{0,1\}^l} \in \{0,1\}^l$ and random $\gamma$-distributed variables $r_{C_{\mathcal{H}}} \in C_{\mathcal{H}}^l$ for any PTIME algorithm $A$.

$\mathsf{Gen}_{C_{\mathcal{H}}}$ takes a bit string as input and generates ciphertext elements. We differentiate two cases, with $\pi$ always being the uniform distribution and $g\colon \{0,1\}^m \to [0,1]^{\hat{m}}$ a PRNG that outputs a sequence of real values in the interval $[0,1]$. The bit string $s \in \{0,1\}^m$ is used as a seed for the pseudo-random generators. The

length of each real value in the output sequence is polynomially-bounded in $m$, as is the length of the output sequence $\hat{m}$. In case $\gamma$ is distinguishable from the uniform distribution, we define the PREG as follows:

$$\mathsf{Gen}_{C_\mathcal{H}}(s) = G_{\approx}(g(s)) \tag{6.3}$$

In the other case, when $\gamma$ is computationally indistinguishable from the uniform distribution, it suffices to let $\mathsf{Gen}_{C_\mathcal{H}}$ expand the input bit sequence using a cryptographically strong $\mathsf{PRBG}\colon \{0,1\}^k \to \{0,1\}^l$ (with $k < l$ and $l$ polynomially-bounded in $k$) and map fresh, unused subsequences of the expanded bit sequence to elements in $C_\mathcal{H}$. Such a mapping is performed sequentially by $\mathsf{BitToElem}\colon \{0,1\}^* \to C_\mathcal{H}^*$, which can be defined as described by Hsieh [Hsi08]. In this second case, $\mathsf{Gen}_{C_\mathcal{H}}$ is defined as:

$$\mathsf{Gen}_{C_\mathcal{H}}(s) = \mathsf{BitToElem}(\mathsf{PRBG}(s)) \tag{6.4}$$

**Proving the generic construction** For now we skip the case that $\gamma$ is indistinguishable from the uniform distribution and prove the generic case. The proof is split into two parts. First we show that a PTIME function to generate $\gamma$-distributed elements in $C_\mathcal{H}$ exists, if there exists an efficient approximation of the quantile function and a secure Pseudo-Random Number Generator (PRNG). This part is later simplified for the special case that $\gamma$ is indistinguishable from the uniform distribution.

Second, we show that if there is a) a PTIME algorithm to generate $\gamma$-distributed values in $C_\mathcal{H}$ for the cryptographic encryption system $\mathcal{H}$, b) its decryption function $\mathsf{Dec}_{\mathcal{H}_\mathsf{sk}}$ is computable in PTIME and c) the homomorphism property of the cryptographic scheme $\mathcal{H}$ holds for all sampled elements in $C_\mathcal{H}$, then a secure Pseudo-Random Key-Stream Generator (PRKG) can be defined.[1]

**Theorem 6.1: Indistinguishable Quantile Function Approx.**

Given a uniform distribution $\pi$, an arbitrary distribution $\gamma$, a uniform random variable $s \in \{0,1\}^m$ and a computationally secure PRBG $g$, if Assumption 6.4.1 holds, it implies a quantile function approximation $G_{\approx}(g(s))$ computationally indistinguishable from $G(u_{[0,1]})$ with $u_{[0,1]} \in \mathbb{R}, 0 \leq u_{[0,1]} \leq 1$ being a uniform random variable.

[1] The BGN system [BGN05] is excluded through the requirement that $\mathsf{Dec}_{\mathcal{H}_\mathsf{sk}}$ is a PTIME algorithm.
Lattice-based crypto is excluded by the requirement that the homomorphism holds for every ciphertext element. Special non-uniform encryptions as used in [KLG13] are supported or excluded depending on the existence of a PTIME algorithm to sample $\gamma$-distributed elements from $C_\mathcal{H}$, such that their decryption yields uniformly distributed elements from $P_\mathcal{H}$.

The triangle inequality over statistical distances is used to derive bounds on computational indistinguishability for two combined computationally indistinguishability results.

*Proof.* Assumption 6.4.1 describes a computationally indistinguishable approximation of the quantile function $G$ given uniform random variables $u_{[0,1]} \in \mathbb{R}, 0 \leq u_{[0,1]} \leq 1$ as input. It follows for any adversary $A$:

$$|\mathsf{P}[G_{\approx}(u_{[0,1]}) = 1] - \mathsf{P}[G(u_{[0,1]}) = 1]| \leq \epsilon_I.$$

From the computationally secure PRBG $g$ follows that an adversary $A'$ is bounded by:

$$|\mathsf{P}[u_{[0,1]} = 1] - \mathsf{P}[g(s) = 1]| \leq \epsilon_g.$$

As $G_{\approx}$ is a deterministic function, it follows for an adversary $A''$ with a maximum runtime identical to the runtime of $A'$:

$$|\mathsf{P}[G_{\approx}(u_{[0,1]}) = 1] - \mathsf{P}[G_{\approx}(g(s)) = 1]| \leq \epsilon_g.$$

The terms of the triangle inequality over statistical distances then allow us to combine both inequalities, that is the probability of distinguishing between $G_{\approx}(u_{[0,1]})$ and $G(u_{[0,1]})$ as given in Assumption 6.4.1, as well as the probability of distinguishing between $G_{\approx}(g(s))$ and $G_{\approx}(u_{[0,1]})$, as given above.

$$|\mathsf{P}[G(u_{[0,1]}) = 1] - \mathsf{P}[G_{\approx}(g(s)) = 1]| \leq \epsilon_g + \epsilon_I.$$

Thus, if there exists a computationally secure PRBG $g$ and a computationally indistinguishable approximation $G_{\approx}$ of $G$, then $G_{\approx}(g(s)$ is a computationally indistinguishable quantile function for $\mathsf{Enc}_{\mathcal{H}}$ with an arbitrary output distribution $\gamma$ and a uniform input distribution $\pi$. □

Using the result of having an indistinguishable approximation of the quantile function give a pseudo-random input, we prove the construction computationally secure regarding the underlying PRBG and call it PREG.

**Theorem 6.2: Secure Pseudo-Random Element Generator**

If $G_{\approx}$ is a PTIME algorithm and a computationally indistinguishable approximation of $G$, and $g$ is a computationally secure Pseudo-Random Number Generator (PRNG) — generating uniformly distributed elements from $\mathbb{R}$ in the interval $[0, 1]$ —, then $\mathsf{Gen}_{C_{\mathcal{H}}} = G_{\approx}(g(s))$ is a computationally secure PREG.

A proof by contradiction is used to show that a computationally secure PRNG combined with a computationally indistinguishable quantile function approximation yields a computationally secure PREG, or otherwise one of the assumptions does not hold.

*Proof.* Assume there exists a PTIME adversary $A'$, which has advantage $\mathsf{Adv}\, A' > e_g$ to distinguish $G_{\approx}(g(s))$ from $G_{\approx}(u_{[0,1]})$, with $u_{[0,1]} \in \mathbb{R}$ being a uniform, independent random variable in the interval $[0,1]$ as given in Section 6.3. Let $A''$ be an adversary, that first evaluates $G_{\approx}$ and then $A'$ in time $t_{\bar{G}} + (t_g - t_{\bar{G}}) = t_g$ upon inputs $g(s)$ and $u_{[0,1]}$. Furthermore, as $G_{\approx}$ is a deterministic function, with no other argument than a (pseudo-)random number, $A''$ also has $\mathsf{Adv}\, A'' > e_g$ in distinguishing $g(s)$ from $u_{[0,1]}$.

Following this, $A''$ would be a distinguisher for $g(s)$ and break the assumption that $g(s)$ is computationally secure. □

We have proven $\mathsf{Gen}_{C_{\mathcal{H}}} = G_{\approx}(g(s))$ to be computationally secure and are going to provide a proof for the second part, the secure PRKG based on the previous result.

**Theorem 6.3: Secure Pseudo-Random Key-Stream Generator**

If $\mathsf{Gen}_{C_{\mathcal{H}}}$ is a computationally secure Pseudo-Random Element Generator (PREG), $\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}$ is a PTIME algorithm, $\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}(\mathsf{Gen}_{C_{\mathcal{H}}}(s))$ generates uniform random values in $P_{\mathcal{H}}$ and the image of $\mathsf{Enc}_{\mathcal{H}_{\mathsf{sk}}}$ is closed under the evaluation function $\mathsf{Eval}_{\mathcal{H}_{\mathsf{sk}}}$ , then $\mathsf{PRKG}_{\mathcal{S}_{\mathcal{H}}}(s) = \mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}(\mathsf{Gen}_{C_{\mathcal{H}}}(s))$ is a computationally secure PRKG for input value $s \in \{0,1\}^*$.

A proof by contradiction is used to show that the decryption function $\mathsf{Dec}_{\mathcal{H}}$ upon indistinguishable input produces computationally indistinguishable output, or otherwise the used PREG is not cryptographically secure.

For ease of presentation and without loss of generality we fix the secret key for a decryption function as $\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}$. This is possible as $\mathsf{sk}$ is independent from the input. The final Pseudo-Random Key-Stream Generator (PRKG) is then described by $\mathsf{PRKG}_{\mathcal{S}_{\mathcal{H}}}(s) = \mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}(G_{\approx}(\mathsf{Gen}_{C_{\mathcal{H}}}(s)))$.

Following the first proof by contradiction for Theorem 6.4.1, the final application of the homomorphic decryption function finishes the construction of a computationally secure PRKG.

*Proof.* Assume that $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}(u_S)$ gives $A$ an advantage $|\mathsf{P}[A(\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}(u_S)) = 1] - \mathsf{P}[A(\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}(u_{C_\mathcal{H}})) = 1]| > e_G + e_{\bar{D}}$ for a $\gamma$-distributed random variable $u_{C_\mathcal{H}}$ to distinguish the outputs, then $\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}$ is a part of an adversary $A'$ that executes first $\mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}$ and then $A$:

$$|\mathsf{P}[A'(G_{\approx}(\mathsf{Gen}_{C_\mathcal{H}}(u_S)) = 1] - \mathsf{P}[A(u_{C_\mathcal{H}}) = 1]| > e_G$$

We therefore found a distinguisher for the PREG $G_{\approx}(\mathsf{Gen}_{C_\mathcal{H}}(u_S))$, which contradicts our assumption about the PREG being computationally secure.

Thus, if there exists a computationally-secure PREG $\mathsf{Gen}_{C_\mathcal{H}}(u_S)$, a computationally indistinguishable approximation $G_{\approx}$ of $G$ and a PTIME algorithm $\mathsf{Dec}_\mathcal{H}$, then $\mathsf{PRKG}_{\mathcal{S}_\mathcal{H}}(u_S))$ is a computationally secure uniform PRKG for values in $P_\mathcal{H}$. □

For the generator to be secure given a public seed $u_S$, it suffices to require $\mathsf{Dec}_\mathcal{H}$ to be computationally hard to be computed without a secret parameter (the secret key $\mathsf{sk}$). This requirement is met by all decryption functions of secure encryption schemes.

### 6.4.2. Analysis of Partly Homomorphic Encryption (PHE) Schemes

Let's consider instantiations of the construction outlined in Section 6.3 with partly homomorphic asymmetric schemes which — upon input from a uniform random variable into $\mathsf{Enc}_\mathcal{H}$ — generate uniformly distributed ciphertexts. The inverse transformation of a uniform random variable is trivially again a uniform random variable and as such the proof of indistinguishability does not depend on the approximate inverse transform assumption. The inverse transformation can in fact be skipped and the PRNG can be applied directly to generate elements uniform in the domain of $\mathsf{Dec}_\mathcal{H}$.

Examples for appropriate PHE schemes are RSA [RSA78], Goldwasser-Micali [GM82], ElGamal [ElG84], Benaloh [Ben94], Naccache-Stern [NS98], Okamoto-Uchiyama [OU98], Paillier [Pai99], Damgard-Jurik [DJ01], BGN [BGN05].

**Theorem 6.4: Secure PRKG from Uniform Ciphertexts**

If $\mathsf{Gen}_{C_\mathcal{H}}$ is a computationally secure PRNG, $\mathsf{Enc}_\mathcal{H}$ generates uniformly distributed values upon uniformly distributed inputs, then $\mathsf{PRNG}_{\mathcal{S}_\mathcal{H}} = \mathsf{Dec}_{\mathcal{H}_{\mathsf{sk}}}(\mathsf{Gen}_{C_\mathcal{H}}(u_S))$ is a computationally secure PRKG.

*Proof.* The proof is identical to the non-uniform $\gamma$-distribution proof for Theorem 6.4.1 given above, without the application of an approximated quantile function and thus without the Assumption 6.4.1. Instead the definition given in Equation (6.4) is used. □

## 6.5. Evaluation

The introduced ciphertext compression scheme is implemented for several PHE systems and evaluated regarding its performance. The selected HE systems are Paillier [Pai99], RSA [RSA78], Goldwasser-Micali [GM82], ElGamal [ElG84], Okamoto-Uchiyama [OU98], Benaloh [Ben94] and Naccache-Stern [NS98].

RSA is deterministic, but nether the less tested, however as it is not an indeterministic cipher, there is no ciphertext expansion and thus no benefit from using compression with respect to transmission and storage efficiency. Also live encryption speed using this scheme is comparable to ECB-mode usage.

Table 6.1 already specifies the factors achieved in terms of storage and communication efficiency. The largest gains are generated for cryptographic systems with a relatively small plaintext group order. An example: choosing Naccache-Stern [NS98] or Benaloh [Ben94] to encrypt 32 bit numbers, and a security parameter of size 1536 bit, we save $1536 - 32 = 1504$ bit on each encrypted value on storage and traffic. That is an increase of about 98% memory and bandwidth efficiency.

Encryption time is also affected, as the encryption function is replaced by the decryption function of $\mathcal{H}$ as shown in Section 6.3. Some cryptographic systems need more time evaluating their decryption function than for their encryption function, *i.e.* [GM82; Ben94; NS98], while others are even faster in decryption [RSA78; ElG84; OU98; Pai99]. Table 6.1 gives average factors for encryption time divided by decryption time under the HE systems. Therefore, if performance is $> 1$, compression speeds up encryption (even without offloading) and vice versa. The given results are mean values over different security parameter settings (between 768 and 4096 bit) and a plaintext group order of size 32 bit (where applicable). Pseudo-random number generation and asymmetric decryption was measured together for the compression scheme encryption performance. As pre-computation of the keystream (offloading) is possible, the actual time evaluating $\mathsf{Enc}_{S_\mathcal{H}}$ is reduced by several orders of magnitude in such a setting.

The initial setup at the server side is also influenced. Namely, the server has to evaluate the $\mathsf{Trans}_{S_\mathcal{H}}$ function before homomorphic operations can be performed, as described in Section 6.3. This includes generating a pseudo-random ciphertext for the homomorphic system $\mathcal{H}$ using the seed $s_0$, encrypting the received value $d_j \in P_\mathcal{H}$ and calculating the homomorphic sum of both values. From these steps, the slowest part is again the homomorphic encryption, which does however not need to be indeterministic — it is enough to perform ciphertext randomization once during output — and can therefore be constructed faster on the server side, than at the client. Furthermore, all plaintext additions, subtractions and inversions can

| Cryptographic Scheme | Encode (ms) | Decode (ms) | Factor |
|---|---|---|---|
| RSA [RSA78] | 133.53 | 34.12 | 3.17 |
| Goldwasser-Micali [GM82] | 0.1 | 2.54 | 0.05 |
| ElGamal [ElG84] | 43.23 | 0.01 | 241.24 |
| Okamoto-Uchiyama [OU98] | 92.59 | 10.94 | 6.91 |
| Benaloh [Ben94] (16 Bit) | 0.65 | 12.13 | 0.05 |
| Naccache-Stern [NS98] (32 Bit) | 1.21 | 85.66 | 0.02 |
| Paillier [Pai99] | 160.35 | 81.15 | 2 |
| JoyeLibert [JL13] (32 Bit) | 1.14 | 6.76 | 0.07 |

Table 6.1.: Measured encryption and decryption times of selected HE systems at 1536 Bit security level. Factor represents encryption divided by time taken for decryption. Factors above 1 describe a speed up, numbers below a slow down when replacing the encryption by the decryption function.

be performed directly on the symmetrically encrypted vales $d_j$ (in their respective plaintext group) before evaluation of $\mathsf{Trans}_{\mathcal{S}_\mathcal{H}}$, leading to possibly higher efficiency gains depending on the protocol.

The performance of this scheme scales much better in the security parameter compared to the related work. This is depicted in Figure 6.2, note that the y-axis is in logarithmic scale and our scheme (without specific performance optimizations for the compression) is at least two orders of magnitude faster than previous ciphertext compression techniques. The perfromance gap between a FHE+AES setup and this scheme grows significantly with an increased security parameter and reaches over three magnitudes of difference at the 80 bit security parameter setting. The relevant literature on FHE AES decryption typically does not give performance values for higher security parameters, as the runtime is much too long for being relevant. Our proposal easily scales to the typical 128 bit security level. Even at this security level, our presented scheme is much faster than previous ciphertext compression schemes at their lowest security setting.

Figure 6.3 presents a few recommendations for the size of keys for asymmetric encryption schemes and its relation to the probably achieved symmetric security eqivalent — *i.e.* a 2000 bits asymmetric key is believed to offer equivalent security as a symmetric key between 90 and 110 bits.

To have a more complete comparison between typically used, applicable AHE schemes and their performance characteristics within *compression*, Figure 6.4, as well as Figure 6.5 and Table 6.1 show what performance to expect from any of these schemes.

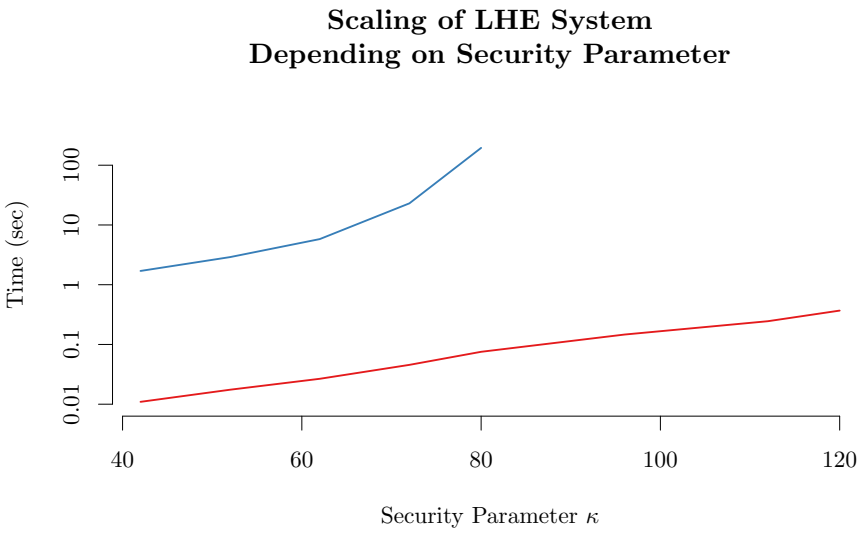



Figure 6.2.: Plot on performance progression with increasing security parameter for best AES decryption performance using Fully Homomorphic Encryption (FHE) and our compression mode as presented in this chapter.

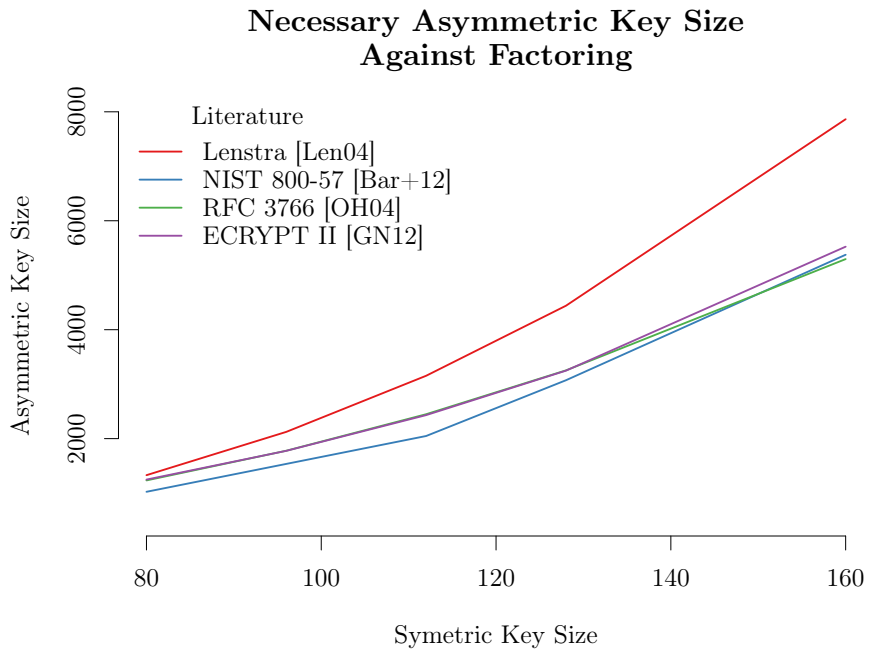



Figure 6.3.: Choice of asymmetric key length based on a desired symmetric security parameter. Recommendations from Lenstra [Len04], NIST Special Publication 800-57 [BKD12], Orman and Hoffman [OH04] and ECRYPT II [GN12]

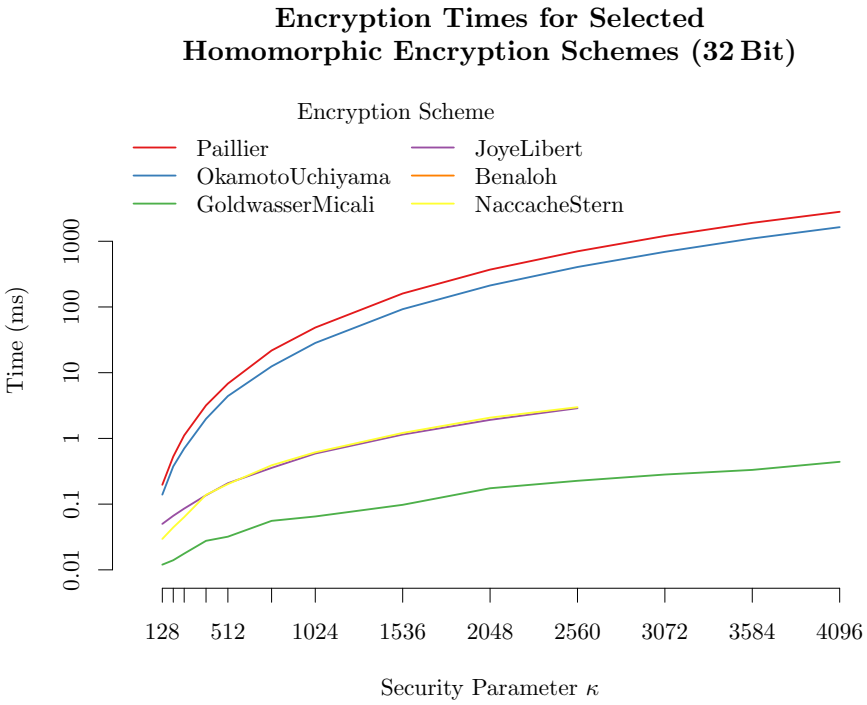



Figure 6.4.: Average Encryption times for a 32 Bit plaintext modules, where applicable. Encryption is performed at the server side and the use of the transition scheme as introduced in Section 6.3

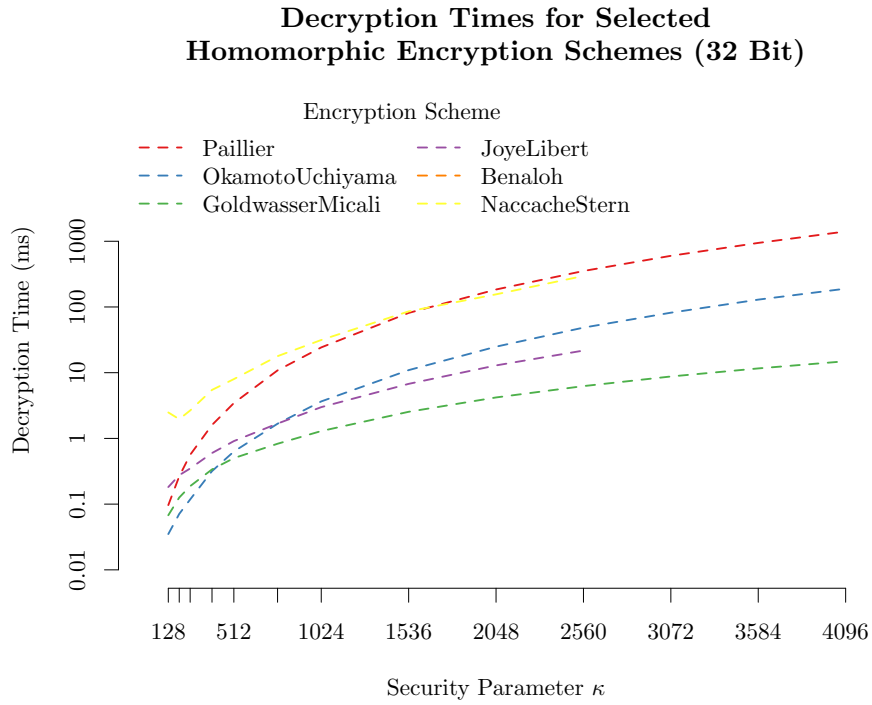



Figure 6.5.: Average Decryption times for a 32 Bit plaintext modules, where applicable. Decryption is performed at the client side and anticipates symmetric encryption under the stream cipher defined in Section 6.3

## 6.6. Conclusion

As semantically secure homomorphic encryption always entails ciphertext expansion due to the included randomness parameter, certain overheads as storage size and increased transmission bandwidth are inherent. We therefore analyzed in this chapter how ciphertext expansion can be reduced or even completely removed. The resulting scheme is a resource-efficient symmetric stream cipher, which offers a highly efficient transition function from the symmetrically encrypted ciphertext to a homomorphically encrypted ciphertext. The symmetric stream cipher is proven secure based on the underlying PRBG and the homomorphic encryption scheme. Furthermore, an experimental evaluation shows the significant performance gains between the transition function of our scheme and the relevant literature, which uses a classical symmetric (lightweight) cipher as transmission cryptographic system and the corresponding decryption function as transition function for homomorphic conversion.

For resource and power-constrained client devices, the option to have pre-computed key-streams for the stream cipher can be important, as no asymmetric operations need to be performed for encryption of input data. The offline generation of the key-stream can be done on less resource-constrained or high-performance personal (trusted) devices, or shared across non-colluding servers through the use of threshold homomorphic encryption schemes.

# 7. Conclusion

The goal of this thesis is to design and analyze a comparison framework under which confidential elements from a certain domain can be compared between two parties. One of the most important requirements for such a framework is the ability to preserve the privacy of user inputs under reasonable assumptions. That is, the use of an external trusted third party is explicitly excluded. Further requirements include the efficient scaling in the size of compared elements (in our case: in the length of the compared strings), an approximate matching for finding similar elements, detection of inference attacks and support for resource-constrained clients.

Before going deeper into the requirements, Section 7.1 will discuss the open research questions, which were explored and for which contributions were published. The discussion showing the fulfillment of the introduced requirements from Section 1.2 follows in Section 7.2. The propositions and results from this thesis are then put into perspective by setting them in relation with other research fields in Section 7.3. Section 7.4 finally discribes possible further research directions for which open questions were identified and thus proposes relevant future work.

## 7.1. Answering Research Questions

Section 1.3 identified several open questions, which guided research like a common thread. These will be recalled and answered using the results from the main chapters of this thesis.

**How can an efficient, privacy-preserving comparison scheme be constructed?** A main endeavor of all of the proposed protocols is to combine privacy-preservation with efficiency and functionality and balance them at a rather high security level. Regarding the achieved privacy and security level, we proved our protocols secure within the Honest-But-Curious (HBC) model, with parts of them even withstanding malicious adversaries — especially when the protocol is enhanced with customized Zero Knowledge Proofs (ZKPs) (Section 5.5). The security proof

of the transition scheme described in Chapter 6, reduces the problem of breaking the proposed scheme to breaking any of the cryptographic primitives used for construction.

As for the efficiency, our protocol scales linearly in the size of the largest input, which is optimal for a secure computation based comparison, for which no information about any (other than its own) input is leaked to the executing party. The constants regarding the performance overhead are also low and the performance evaluations in the main chapters 4,5,6 show the practicability of the proposed protocols.

The combination of described techniques within the previous chapters leads to schemes which offer a better performance compared to proposed protocols from the related work, while offering a similar or higher security and feature level.

**How can an inference attack be detected and limited, given encrypted inputs?** Inference attacks typically try to interleave related information gathered about confidential data in order to reconstruct parts or even all of it. In a privacy-preserving comparison scheme such an attack could be performed by requesting a larger number of comparisons against the same element and use the returned similarities to narrow down the number of possible elements. Of course, a party wants to limit (in terms of detect and mitigate) such an inference attack to control leakage of confidential information.

This issue is approached in two different ways. First, the extended protocol in Chapter 4 reduces the amount of returned information as much as possible — only a single bit of output can be returned as the result of a comparison. Second, a mechanism that detects similar queries regarding some metric is used to detect and prevent too close queries from being answered. As a result, inference attacks are much less effective.

By using fuzzy commitments together with error-correcting codes from coding theory, the detection can be efficiently performed with minimal information leakage towards the server. The only leakage that occurs is whether a similar request was seen before or not, which is optimal for an efficient inference detection scheme that does plaintext equation testing. We gave a construction of such a scheme in Chapter 5 together with a protocol secure against malicious users. A thorough security analysis, involving novel ZKPs and an extensive evaluation on how such a scheme could be instantiated is presented. Furthermore, its effectiveness is studied analytically and emperically using a database composed of human mitochondrial genomic sequences.

**How can a client reduce cryptographic overheads?** Homomorphic cryptographic schemes are typically based on asymmetric cryptography and therefore inherit also their drawbacks. Rather low performance and ciphertext expansion are two of them, which affects especially small, mobile devices. With limited amounts of main memory and computational capacity it might be infeasible to perform expensive asymmetric, homomorphic cryptographic operations — typically over large prime- or composite-order groups — upon them. Therefore we present a scheme that allows a mobile client to store a pre-computed keystream for symmetrical on-the-fly encryption of data in Chapter 6. The special property of this keystream lies in the way it was generated. The result of this construction is a trivially convertible stream cipher, which can be very efficiently transformed into an homomorphic encryption of the same data without the need for plaintext decryption.

Therefore the client does not need to perform expensive asymmetric operations, reducing its power drain and latency. Furthermore, ciphertext expansion that is introduced using the asymmetric cryptographic schemes is removed, which also reduces the transmission size, time and again the necessary power for the radio frequency transmission. This scheme improves performance over similar current schemes by several orders of magnitude and is the first symmetric to homomorphic encryption transition scheme which can be practically used. Even more, the related work on ciphertext compression is restricted to rather low security levels typically between 56 bit and 80 bit security due to their impractical performance. Our scheme can easily use typical security levels of 128 bit security and more.

## 7.2. Fulfillment of Requirements

Chapter 1 introduced a detailed problem description together with a desired solution — namely a comparison scheme, which should fulfill certain requirements. These requirements are recalled and discussed regarding their achievement.

**Privacy-Preservation** One of the most important design goals of the comparison scheme is to preserve the input privacy of both parties. As such, the scheme should neither require a trusted third party to protect confidentiality of inputs, nor rely on obfuscation or anonymization techniques, which may give only weak privacy guarantees. Following these requirements, unnecessary leakage of information is equally unwanted. As a result the server should learn nothing about the input from the client, while the client only learns the computed distance. However, for efficiency and thus practicability reasons the adversary is said to follow the HBC model, as introduced in Section 1.5.1.

The security and privacy related part of the basic schemes presented in Chapter 4 is analyzed in Section 4.3 and shown to follow the guarantees of the underlying cryptographic homomorphic encryption scheme. The extension towards a privacy-preserving detection of inference attack queries presented in Chapter 5 also has a thorough security analysis in Section 5.6. It further includes a novel ZKP constructions to enable the usage of the inference control framework when a malicious client is involved. In this extension the server has to get the information whether two strings are similar or not to be able to make decisions based on this. As the server learns something about the input of the client, the possible leakage is analyzed in Section 5.6. It was shown that the only leaked information is if queries are pair-wise close to each other, nothing else. In particularly no distance is leaked.

Last but not least, the extension to reduce transmission and storage bandwidth in Chapter 6 features a detailed security analysis, which proves the introduced stream cipher as secure as the underlying cryptographic system and the Pseudo-Random Number Generator (PRNG). It does not leak any information to the server.

Following these analyses, the overall framework incorporating the inference control and transmission efficiency extension, gives very high input privacy guarantees for both participating parties. Furthermore, the confidentiality of the client input is even preserved in the presence of an malicious server.

**Efficiency for Long Strings** The design goal of input privacy can be achieved using standard building blocks for the complete functionality, like Secure Multi-Party Computation (SMPC) or homomorphic encryption. Such generic construction can also include a similar inference control algorithm as presented in Chapter 5. However, the resulting scheme will not be practical, as secure dynamic programming and inference control inside secure computation are both inherently inefficient and lead to protocols, which do not scale (see Section 2.1.2 and 5.1).

Therefore an embedding from the string edit distance into Hamming space is used to reduce the quadratic complexity of the edit distance dynamic programming algorithm to the linear complexity of calculating the xor-cardinality of two binary vectors. The solution further involves only low constant overhead and was evaluated using human mitochondrial DNA sequences, which could be compared in the order of about 1 minute on commodity hardware.

As long as the HBC model is used, also the inference control extension describes a very efficient scheme for detecting and mitigating inference attacks. However, once the client is not believed to follow the HBC model anymore, a rather large constant overhead is introduced by the need to validate the clients input and computation. The complexity of checking the commitment in order to detect an

inference attack, is in the naïve setting linear in the number of previous requests, but can be performed via a hashset lookup having complexity $\mathcal{O}(1)$.

Following the motivation and design goals, the extension for higher transmission and storage efficiency is of course especially designed with efficiency in mind. It allows for faster transmission of queries to the server and a very simple and fast conversion of the transmitted data into homomorphic encryptions on the server side. The total workload of the client might be increased by the use of the asymmetric decryption instead of the encryption function, but then again decryption can be outsourced to be either pre-computed or being shared using a threshold encryption scheme — or both.

Experimental evaluations support the theoretical complexity analysis in the different chapters. Further related to the efficiency of the protocol is the choice of the homomorphic encryption scheme. All presented protocol variants, extensions and algorithms only need an Additive Homomorphic Encryption (AHE) scheme, except for the ZKPs in Chapter 5 which will work given the efficient BGN system [BGN05]. So the most performant additive scheme can be picked. Regarding the transmission efficiency of the proposed stream cipher in Chapter 6, systems with a rather small plaintext space are preferred over systems having a large plaintext group size.

The protocols mainly work on vectors within Hamming space, but also sum up combinations of them. As the proposed Bloom filter length $n$ for the discussed use case of comparing mitochondrial genomes is $2^{14} < n < 2^{15}$, which equals the dimension of the used Hamming space, summing up all elements of combinations of such vectors in euclidean space can never exceed $n$. Therefore the plaintext domain can be restricted to contain up to $2^{15}$ elements without overflowing. Consequently the plaintext is represented by 15 Bit integers, allowing for large gains in transmission efficiency, *i.e.* if the cryptographic scheme by Joye and Libert [JL13] is used.

**Approximate Matching** Data stored in different places might not always be accurate or identical due to sequencing errors for genomic data, typographical errors or spelling mistakes via manual input or data corruption during transmission. Out of this follows the requirement to not only match identical elements, but also calculate distances between them and possibly return if they are in a certain range.

The two different meanings of "approximate matching" in the relevant literature are both covered. One usage is to find elements, which have a low distance regarding some metric. The other meaning simply implies calculating and returning a distance between two elements. The algorithms specified in Chapter 4 cover both

definitions. The basic comparison protocol depicted in Figure 4.1 returns the difference between compared elements, while the extended protocol (see Figure 4.2) returns whether both compared elements are within a certain distance range. The approximate matching property of the protocol versions introduced in Chapter 4 is not influenced by the extensions described in Chapters 5 and 6.

The actual distance computation is solved via an approximation by embedding the edit distance into Hamming space and have the Hamming distance returned. This embedding is not isometric, as there exists no isometric embedding from the edit distance metric to a metric over the Hamming space. From this follows that the embedding introduces distortion, which leads to the field of fuzzy matching. Chapter 4 shows a low distortion during the empirical evaluation, especially for comparisons of low-distance sequences.

**Non-Interactive Protocol** Designing a protocol that has a high degree of interactivity between participating parties most certainly introduces network overheads due to latency or time used for transmission. It further makes it harder to use the protocol when the connection between both parties is unstable or has a high latency. This might be true when the client is a mobile, resource- and power-constrained device, which uses radio communication to connect to the other party. Another positive effect of a non-interactive protocol is that if the execution of the protocol takes a rather long time, all parties can perform all necessary calculations offline after the initialization of the protocol and upload the results for later retrieval. In such a protocol both parties must never be online at the same time, as also the protocol initialization can be done asymmetrically.

All of the presented protocols are non-interactive and can be easily adopted to use an always online storage system like a bulletin board to post comparison requests and results in a completely asymmetric way. The only protocol that doesn't work like this is the inference control protocol under the assumption of a malicious user. In such a case the protocol will be interactive by requiring several Zero Knowledge Proofs (ZKPs) between the parties.

**Two-Party** If a protocol that uses input from two parties also only requires those two to calculate the desired result, then there is no need to care about any other party, its intentions or whether it might collude with one of the two input parties. All our protocols work in such a setting and never require an external party to join the protocol. However, a party might outsource calculations and data to any external party as he likes. This case is covered by the definition of the Honest-But-Curious (HBC) adversary model.

**Resource-Constraint Clients** As the Internet of Things (IoT), together with ubiquitous computing and Cyber-Physical Systems (CPSs) gains traction and spawns (as well as fuels) areas like mobile sensor networks, mHealth[1], smart homes and wearables, we are going to look at and deal with large quantities of small, mobile, battery-powered devices. These devices must be as power efficient as possible, which includes controlled and minimized use of the compute capacity, as well as limited use of radio-frequency communications.

Out of these constraints evolves a transmission efficient protocol extension as described in Chapter 6. It allows to send encrypted data only expanded to the size of the plaintext domain, not the size of the ciphertext domain. Following the example given in the efficiency paragraph above: using a plaintext size of 16 bits suffices for computing the desired functionality, while expanding all homomorphically transmitted plaintexts to these 16 bits instead of the typical ciphertext sizes between 1536 and 3072 bits. As such, two orders of magnitude in communication size can be saved for the discussed example.

Furthermore, utilizing the pre-computation and outsourcing the offline computation step at the client side to more powerful devices decreases encryption latency, live encryption time and power usage at the thin client notably.

**Size-Hiding** The system comparison parameters are chosen in the beginning independent from the actual input of the participating parties. The size of the private input, the string length in our case, is equally well handled. None of the algorithms and protocols makes assumptions about the length of the input and thus they also do not use this parameter inside the proposed schemes.

**Inference Control** No related work exists on performing inference detection and control upon encrypted or otherwise secured data, so it is not possible to evaluate against existing solutions. However, inference attacks are simulated by issuing low distance requests and evaluating the probability to detect them as similar — which finally leads to a detection and rejection. It was shown in the evaluation of Chapter 5 that the probability of detecting two queries as similar depending on the distance between them, renders a steep curve, which nicely separates pairwise distances with a high detection probability from distances that show a low detection probability.

[1] mobile eHealth: describing the remote observation and possibly limited remote treatment of patients

## 7.3. Relation to Other Research Fields

Starting with the basic comparison schemes described in Chapter 4, an efficient string comparison scheme was developed, analyzed and evaluated. Of course the scheme as it was presented can easily be adopted to compare elements from other domains — replacing the strings or sequences. This includes using the underlying homomorphic scheme directly to compare and compute the Manhattan distance ($l_1$-norm). This can be valuable as finding an embedding into $l_1$ might be easier or better described than finding an embedding into Hamming space. Arbitrary other metrics that have embeddings into the Manhattan or Hamming distance can therefore be effectively approximated using to described schemes.

The work on inference control over encrypted input data is of independent interest and can be used for arbitrary other privacy-preserving protocols, which allow approximations of element distances to steers inference control algorithms. The detection scheme is similarly flexible in the choice of actual elements and used embeddings, as the comparison scheme is. Furthermore, the presented ZKPs cover a proof for correct error-correction and decoding of the Reed-Muller decoding equations, which might also come handy for other protocols that include Error-Correcting Codes (ECCs) within the malicious or covert adversary model. An example would be a malicious decoder of a noisy channel, which sends corrected and decoded messages homomorphically encrypted to further recipients.

Similarly, the presentation of a highly efficient transition scheme from a symmetric to an homomorphic scheme is also of independent interest, as this extension can be used to potentially speed up many privacy-preserving protocols that utilize transmission or storage of homomorphically encrypted information.

## 7.4. Future Work

The previous chapters described a comparison scheme that is efficient for long input strings, preserves a high level of privacy, can identify inference attacks, can be made efficient for resource-constraint clients and fulfills also all the other requirements mentioned in Section 1.2. However, once a comparison scheme exists to compare one element to another and evaluate its similarity, it will be used to compare an element against many others and select a desired element depending on the resulting similarity. When this comparison scheme is thus used for searching a large database, it has to be applied once for every entry in the database resulting in a search time linear in the size of the database, which does not scale for very large databases.

To actually perform a search more efficiently, some requirements emerge. First of all, the search itself must obviously be more efficient than the one-to-all comparison sketched above. It must also not leak information about the actual query, as such leakage would defeat the strong security and privacy properties of the presented comparison scheme. A notion for the probability of false-positives and false-negatives similar to Locality-Sensitive Hashing (LSH) should be described (Section 2.1.1).

Such a search scheme could make use of an index built over the database, which groups different entries together regarding some properties or distance. A "privacy-preserving set intersection with data transfer" protocol would then reveal which entry IDs represent entries, which are most similar to the own element. A detailed comparison between those elements and the own one could then be performed using our proposed comparison scheme.

# A. Edit Distance Embedding

## A.1. VGRAM Intersection Cardinality

The string distance approximation scheme proposed in Chapter 4 describes the use of set operation. Specifically, strings are transformed into a set and the difference set ($A \oplus B = (A \cup B) \setminus (A \cap B)$ — the union minus the intersection for two sets $A, B$) is used to approximate the edit distance. Another primitive set operation to approximate the distance would be to simply take the set intersection as a measure for similarity and transform it into a distance.

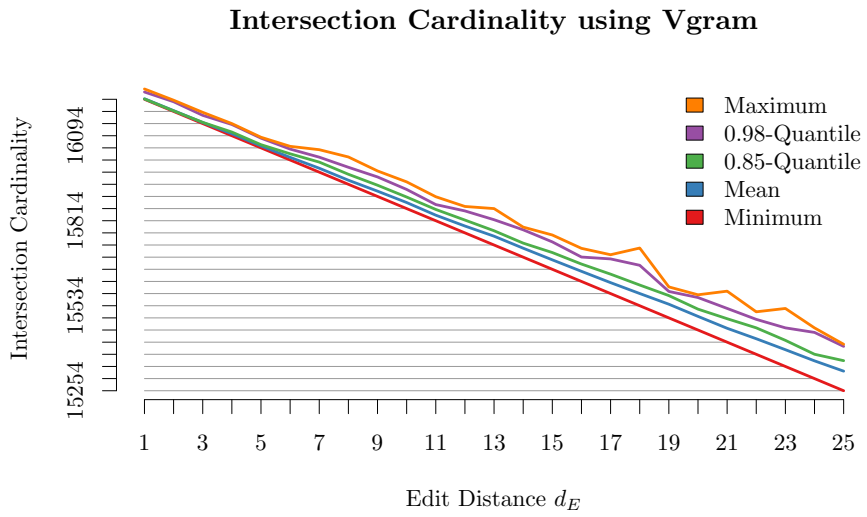



Figure A.1.: Intersection cardinality between sets of variable length grams generated out of genomic sequences from the "Human Mitochondiral Genome Database" [IG06] using the VGRAM algorithm [LWY07]. Minimum, Mean, Maximum, as well as 0.85- and 0.98-Quantiles for the intersection cardinality are plotted depending on the edit distance between the underlying genomes.

Figure A.1 plots the measured intersection cardinality for human mitochondrial genomic sequences depending on the edit distances. Mininum, maximum, mean,

0.85- and 0.98-quantile values are given. One can especially see that the minimum values titled "lower bound" in Figure A.1 closely resembles a linear function, which can be useful in cases edit distances above some threshold need to be excluded.

## A.2. Edit Distance Prediction

Another, related issue regarding the edit distance approximation described in Section 4.2 and A.1 is the prediction of the edit distance between two character sequences given their VGRAM [LWY07] (see Section 3.3) intersection cardinality. We test the accuracy of predicting the edit distance between two strings by building a table $T$ that stores the mean intersection cardinality between VGRAM sets for a specific edit distance. Let $T_{d_E}$ denote the mean intersection cardinality for edit distance $d_E$. For this purpose we generate 100 pairs of character sequences for each edit distance $d_E$, compute their VGRAM sets and finally the intersection cardinality $c$ upon them. For prediction we select the closest $t = T_{d'_E} \in T$ such that the difference between $c$ and $t$ is minimal: $\min_{t \in T} |c - t|$. Let $d'_E$ be the predicted edit distance given the intersection cardinality $c$ and let $|x|$ be the absolute value of $x$. Following this we define the prediction error in terms of edit distance offset: $d'_E - d_E$. Figure A.2 depicts for all evaluated edit distances $d_E$ the percentage of an edit distance offset that occurred during prediction.

Negative error values are thus predicted edit distances, which are smaller than the correct edit distance between the compared character sequences.

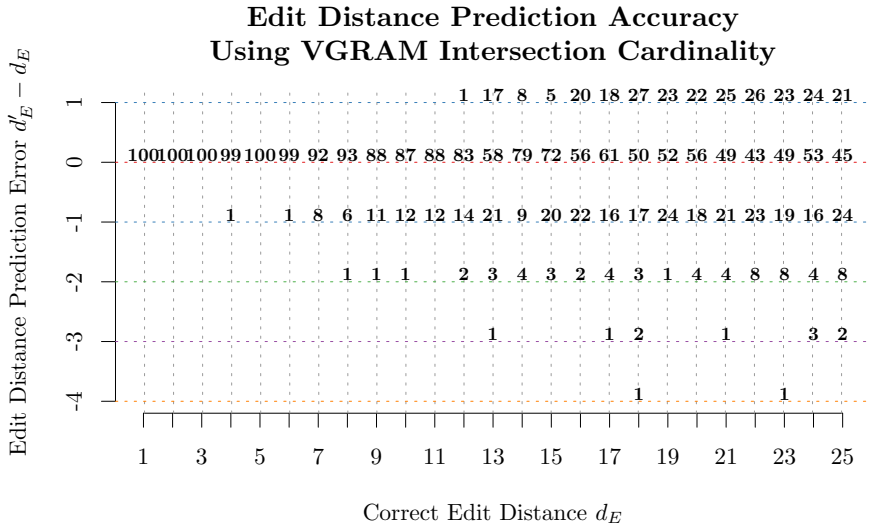



Figure A.2.: Percentages of edit distance errors that occurred during prediction of the edit distance between two character strings given their VGRAM intersection cardinality. The mean intersection cardinality over VGRAM sets for a specific edit distance $d_E$ is used for selecting the predicted edit distance $d'_E$.

## A.3. Bloom Filter False-Positive Rate

The accuracy of approximating the edit distance is slightly affected by the choice of false-positive rate. Figure 4.3 shows the difference between a configuration $p_{\text{fp}} = 0.1$ and $p_{\text{fp}} = 0.5$ in terms of resulting Hamming distance given a certain edit distance. Figure A.3 makes the difference between the configurations more explicit, by plotting the ratio between Hamming distances in both configurations. Basically the Hamming distance results are compressed by a factor $\approx 1.7$ for $p_{\text{fp}} = 0.5$, which means that adjacent Hamming distance distributions, as depicted in Figure 4.4 are a bit less overlapping for $p_{\text{fp}} = 0.1$ than shown in the actual Figure 4.4. The pearson correlation between the edit distance of the original strings and our distance measure remains also very high at $c_p = 0.997$ for $p_{\text{fp}} = 0.5$.

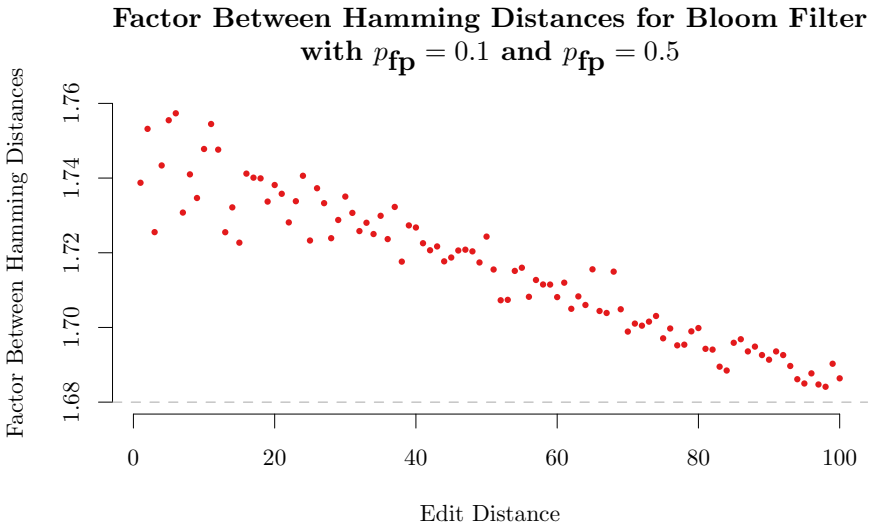



Figure A.3.: Factor between distances generated by the basic protocol described in Chapter 4. There is a clear trend towards a lower factor for higher edit distances. The pearson correlation between the factors and the edit distance is $c_p = -0.957$.

# B. Configuration of the Error-Correcting Code

Chapter 5 proposes to detect similar inputs by having a function that produces collisions given similar input. Similarity is defined by the Hamming distance on bit strings. Mapping the elements of the actual comparison to bit strings is not part of this chapter, but was discussed for character strings — especially genomic sequences — in Chapter 4. The task of producing collisions depending on the Hamming distance of bit strings is proposed to be solved using Error-Correcting Codes. Specifically the Reed-Muller [Mul54; Ree54] codes are interesting, because they use a simple decoding function and inhibit a strong error-correction capability. Using this code family, we choose to only select first-order codes represented as $\mathcal{R}(1, m)$, due to their linear decoding equations and simple structure. The codes can be completely specified with a single parameter $m$. This parameter influences the length of the information and codewords, as well as the overall capabilities of the code. Selecting different values for $m$ will result in different error correction and therefore inference detection performance. The next sections will present results on different possible choices of the parameter $m$.

## B.1. Bloom Filter Mapping Depending on Reed-Muller Configuration

Selecting a specific value for $m$ fixes the length of the codewords, as well as the length of the information words generated by decoding codewords. As the input bit strings for the Error-Correcting Codes (ECCs) are in their length more or less independent of the configuration of the ECC, we can calculate the ratio between the codeword length and the input bit string length. Also depending on the actual configuration are the number of correctable bits and other variables. The generated codes have codeword lengths which are only a fraction of the input length, an thus need to be applied several times to use and decode as much as possible from the input. The use of several identical instantiations of an ECC scheme is called a "combined" ECC scheme.

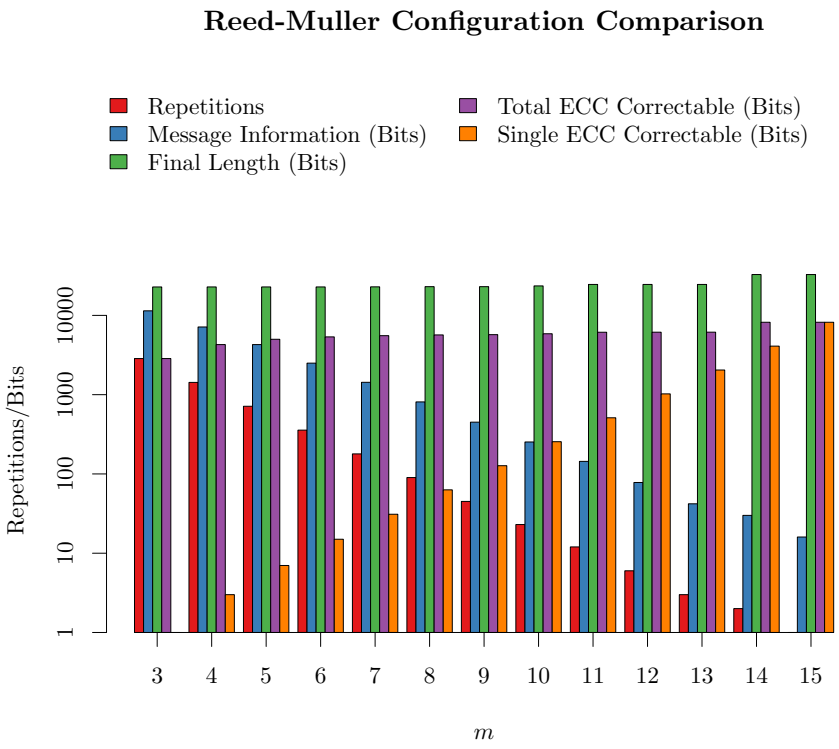



Figure B.1.: Properties of combined Reed-Muller schemes to reach a total length close to 22836 bits.

Figure B.1 gives properties of combined ECC schemes to reach a minimum combined length of 22836 bits, as estimated in Chapter 5. "Repetitions" denotes the number $r$ of codes combined to reach the overall length. "Message Information (Bits)" describes how many bits can be distinguished after decryption of all ECCs, this equals the length of the information word times the number of ECCs repetitions $r$: $(m+1) \cdot r$. "Final length (Bits)" equals the codeword length times the repetitions $2^m \cdot r$, "Single ECC correctable" presents the performance of a single Reed-Muller code with configuration $m$, while "Combined ECC correctable" represents the maximal performance of the combined scheme.

It can be seen easily that with larger values for $m$ the number of codes (repetitions $r$) combined to decode the maximum amount of input bits gets smaller, while obviously the number of bits decoded per ECC increases. However, it can also be seen that the overall number of usable bits stays in the same order of magnitude, similar to the number of total correctable bits over all ECC repetitions.

## B.2. Decoding Probability Depending on Reed-Muller Configuration

In Section 5.7.4 we presented a framework for analytic evaluation of the first-order Reed-Muller ECC given different values for $m$, the mapping offset $o$ introduced in Section 5.8.2, as well as different Hamming distance between two inputs. The probability of decoding both input bit strings to the same value is analytically evaluated and plotted in Figure B.2.

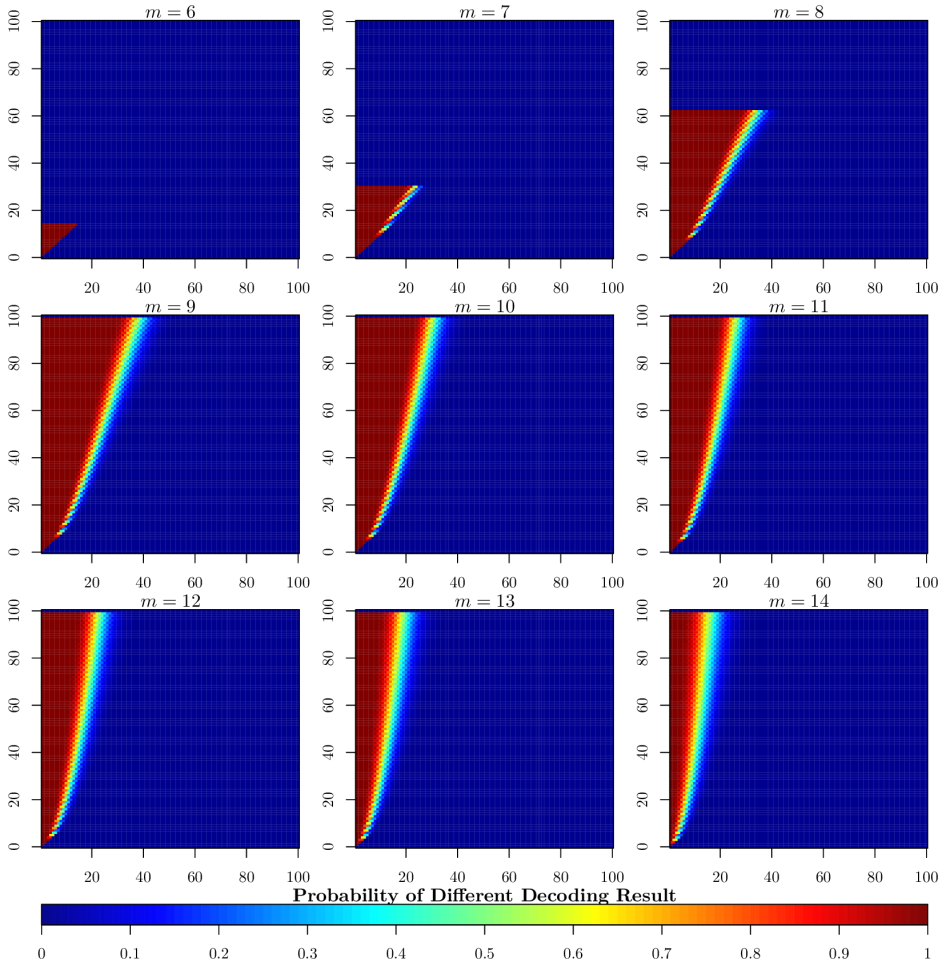



Figure B.2.: **Theoretical analysis:** Probabilities to decode two different Bloom filters into different commitments using the Reed-Muller codes $\mathcal{R}(1, m)$ for $m \in [6, 14]$. Probabilities depend on the Hamming distance $k$ between the Bloom filters, as depicted on the $y$-axis of the plots and the offset $o$ of the initial Bloom filter from a codeword, depicted on the $x$-axis.

Next to the analytic analysis using a description of the Reed-Muller decoding function, we performed an empirical evaluation actually instantiating first-order Reed-Muller codes with different parameters, applying close-to-codeword mappings given different values for $o$ and using different Hamming distance input strings. The results are plotted in Figure B.3.

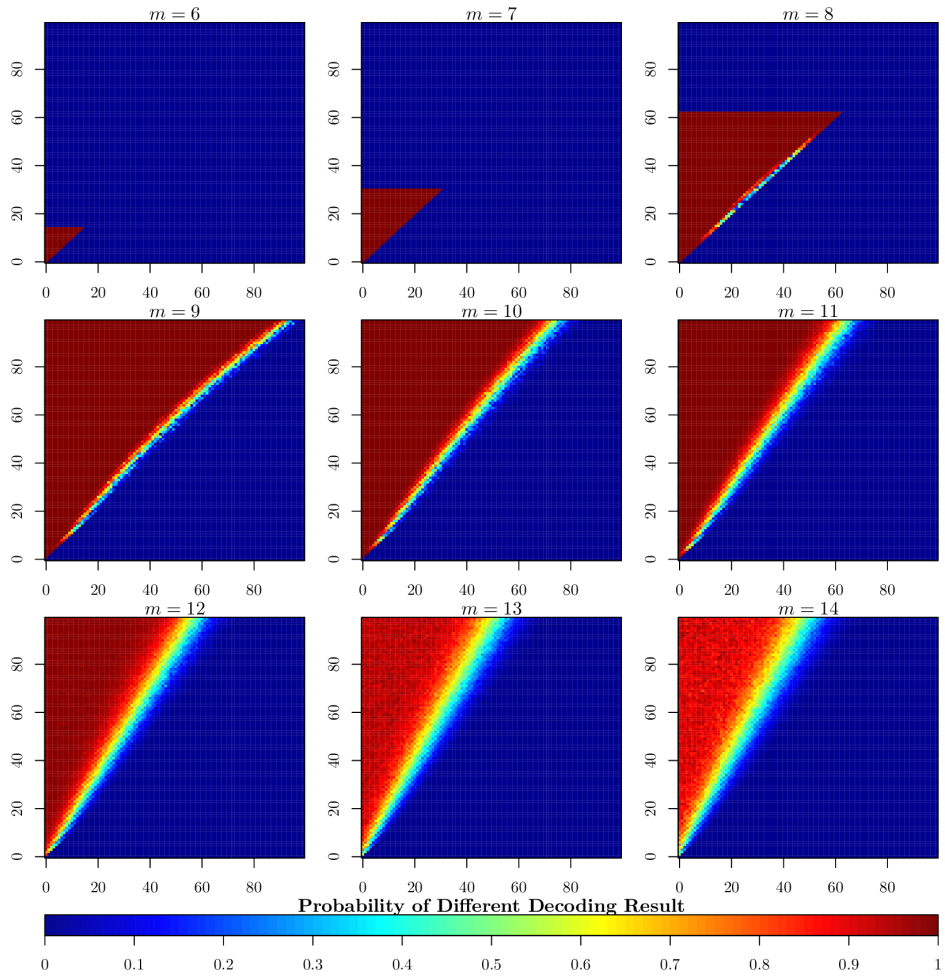



Figure B.3.: **Empirical analysis:** Probabilities to decode two different Bloom filters into different commitments using the Reed-Muller codes $\mathcal{R}(1, m)$ for $m \in [6, 14]$. Probabilities depend on the Hamming distance $k$ between the Bloom filters, as depicted on the $y$-axis of the plots and the offset $o$ of the initial Bloom filter from a codeword, depicted on the $x$-axis.

# Acronyms

| | |
|---|---|
| **ABE** | Attribute-Based Encryption . . . . . 46 |
| **AES** | Advanced Encryption Standard . . . . . 140 |
| **AHE** | Additive Homomorphic Encryption . . . . . 163 |
| **ASIC** | Application-Specific Integrated Circuit . . . . . 53 |
| **BDSG** | Bundesdatenschutzgesetz . . . . . 4 |
| **CA** | Certificate Authority . . . . . 28 |
| **CDF** | Cumulative Distribution Function . . . . . 146 |
| **CPS** | Cyber-Physical System . . . . . 165 |
| **ECC** | Error-Correcting Code . . . . . 173 |
| **ESE** | Efficiently Searchable Encryption . . . . . 45 |
| **EXPTIME** | deterministic exponential time . . . . . 20 |
| **FDPA** | Federal Data Protection Act . . . . . 4 |
| **FE** | Functional Encryption . . . . . 46 |
| **FHE** | Fully Homomorphic Encryption . . . . . 94 |
| **FPGA** | Field-Programmable Gate Array . . . . . 53 |
| **FSM** | Finite State Machine . . . . . 56 |
| **HBC** | Honest-But-Curious . . . . . 159 |
| **HE** | Homomorphic Encryption . . . . . 139 |
| **HIBE** | Hierarchical Identity-Based Encryption . . . . . 47 |
| **i.i.d.** | independent and identically distributed . . . . . 141 |
| **IBE** | Identity-Based Encryption . . . . . 46 |
| **IND-CPA** | Indistinguishability under Chosen Plaintext Attack . . . . . 94 |
| **IND-CKA** | Semantic Security Against Adaptive Chosen Keyword Attack . . . 41 |
| **IoT** | Internet of Things . . . . . 165 |
| **KDF** | Key Derivation Function . . . . . 23 |
| **LHE** | Leveled Homomorphic Encryption . . . . . 61 |
| **LSH** | Locality-Sensitive Hashing . . . . . 167 |
| **M2M** | Machine-to-Machine . . . . . 1 |
| **MAC** | Message Authentication Code . . . . . 20 |
| **MHE** | Multiplicative Homomorphic Encryption . . . . . 141 |
| **MITM** | Man-In-The-Middle . . . . . 70 |
| **MPC** | Multi-Party Computation . . . . . 54 |
| **mtDB** | Human Mitochondrial Genome Database . . . . . 13 |
| **NP** | non-deterministic polynomial time . . . . . 20 |
| **OFSM** | Oblivious Finite State Machine . . . . . 37 |
| **OPE** | Oblivious Polynomial Evaluation . . . . . 29 |

**OPRF** Oblivious Pseudo-Random Function . . . 29
**ORAM** Oblivious RAM . . . 37
**OSI** Open Systems Interconnection . . . 23
**OT** Oblivious Tranfer . . . 29
**OWF** One-Way Function
**PDP** Proof of Data Possession . . . 48
**PEKS** Public key Encryption with Keyword Search . . . 40
**PHE** Partly Homomorphic Encryption . . . 141
**PII** Personally Identifiable Information . . . 1
**PIM** Private Information Matching . . . 39
**PIMPD** Private Information Matching from Public Database . . . 39
**PIR** Private Information Retrieval . . . 29
**PKI** Public Key Infrastructure . . . 46
**PoR** Proof of Retrieval . . . 48
**PPDP** Privacy-Preserving Data Publishing . . . 49
**PRF** Pseudo-Random Function . . . 44
**PRP** Pseudo-Random Permutation . . . 43
**PRBG** Pseudo-Random Bit Generator . . . 140
**PREG** Pseudo-Random Element Generator . . . 143
**PRKG** Pseudo-Random Key-Stream Generator . . . 143
**PRNG** Pseudo-Random Number Generator . . . 162
**PSI** Private Set Intersection . . . 28
**PSI-CA** Private Set Intersection Cardinality . . . 28
**PSU** Private Set Union . . . 28
**PSS** Private Similarity Search . . . 38
**PTIME** deterministic polynomial time . . . 141
**rCRS** revised Cambridge Reference Sequence . . . 74
**ROM** Random Oracle Model . . . 29
**SC** Secure Computation . . . 39
**SDP** Secure Dynamic Programming . . . 36
**SE** Searchable Encryption . . . 41
**SHE** Somewhat Homomorphic Encryption . . . 94
**SIMD** Single Instruction Multiple Data . . . 51
**SMPC** Secure Multi-Party Computation . . . 162
**SSCO** Secure Storage and Computing Outsourcing . . . 40
**SSE** Symmetric Searchable Encryption . . . 40
**SSO** Secure Storage Outsourcing . . . 40
**TTP** Trusted Third Party . . . 33
**UPF** Unpredictable Function . . . 29
**VoD** Video on Demand . . . 1
**ZKP** Zero Knowledge Proof . . . 159

# Index

# Bibliography


[Abe+12] Goncalo R Abecasis et al. "An integrated map of genetic variation from 1,092 human genomes." In: *Nature* 491.7422 (Oct. 2012), pp. 56–65. DOI: 10.1038/nature11632. URL: http://www.nature.com/nature/journal/v491/n7422/full/nature11632.html (visited on 06/11/2015).

[AFT06] Tatsuya Akutsu, Daiji Fukagawa, and Atsuhiro Takasu. "Approximating Tree Edit Distance Through String Edit Distance." In: *Algorithmica* 57.2 (June 2006), pp. 90–99. ISSN: 1432-0541. DOI: 10.1007/s00453-008-9213-z. URL: http://link.springer.com/10.1007/s00453-008-9213-z (visited on 06/11/2015).

[Agh+14] Armen Aghasaryan et al. "Exploring the impact of LSH parameters in privacy-preserving personalization." English. In: *Bell Labs Technical Journal* 18.4 (Mar. 2014), pp. 33–44. ISSN: 1089-7089. DOI: 10.1002/bltj.21644. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6770346 (visited on 06/11/2015).

[AI09] Nuttapong Attrapadung and Hideki Imai. "Conjunctive broadcast and Attribute-based encryption." English. In: *Proceedings Third International Conference on Pairing-Based Cryptography (Pairing 2009)*. Vol. 5671. LNCS. Palo Alto, CA, USA: Springer Berlin Heidelberg, 2009, pp. 248–265. ISBN: 978-3-642-03298-1. DOI: 10.1007/978-3-642-03298-1_16. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-03298-1_16 (visited on 06/11/2015).

[AKD03] Mikhail J. Atallah, Florian Kerschbaum, and Wenliang Du. "Secure and private sequence comparisons." English. In: *Proceeding of the ACM workshop on Privacy in the electronic society (WPES '03)*. Washington, DC, USA: ACM Press, Oct. 2003, pp. 39–44. ISBN: 1-58113-776-1. DOI: 10.1145/1005140.1005147. URL: http://dl.acm.org/citation.cfm?doid=1005140.1005147 (visited on 06/11/2015).

[AKP13] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. "Group Homomorphic Encryption: Characterizations, Impossibility Results, and Applications." In: *Des. Codes Cryptography* 67.2 (2013), pp. 209–232. ISSN: 0925-1022. DOI: 10.1007/s10623-011-9601-2. URL: http://dx.doi.org/10.1007/s10623-011-9601-2 (visited on 12/18/2015).

[Aku06] Tatsuya Akutsu. "A relation between edit distance for ordered trees and edit distance for Euler strings." In: *Information Processing Letters* 100.3 (Nov. 2006), pp. 105–109. DOI: 10.1016/j.ipl.2006.06.002. URL: http://www.sciencedirect.com/science/article/pii/S0020019006001761 (visited on 06/09/2015).

[AL05] Mikhail J. Atallah and Jiangtao Li. "Secure outsourcing of sequence comparisons." English. In: *International Journal of Information Security* 4.4 (Mar. 2005), pp. 277–287. ISSN: 1615-5270. DOI: 10.1007/s10207-005-0070-3. URL: http://link.springer.com/article/10.1007%2Fs10207-005-0070-3 (visited on 06/11/2015).

[AL09] Yonatan Aumann and Yehuda Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries." English. In: *Theory of Cryptography* 23.2 (Apr. 2009), pp. 281–343. ISSN: 1432-1378. DOI: 10.1007/s00145-009-9040-7. URL: http://link.springer.com/10.1007/s00145-009-9040-7 (visited on 06/11/2015).

[Alb+15] Martin Albrecht et al. "Ciphers for MPC and FHE." English. In: *Proceedings 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015)*. Vol. 9056. LNCS. Sofia, Bulgaria: Springer Berlin Heidelberg, Apr. 2015, pp. 430–454. ISBN: 978-3-662-46800-5. DOI: 10.1007/978-3-662-46800-5_17. URL: http://link.springer.com/chapter/10.1007%2F978-3-662-46800-5_17 (visited on 06/11/2015).

[And+81] S. Anderson et al. "Sequence and organization of the human mitochondrial genome." In: *Nature* 290.5806 (Apr. 1981), pp. 457–465. DOI: 10.1038/290457a0. URL: http://www.nature.com/nature/journal/v290/n5806/abs/290457a0.html (visited on 06/11/2015).

[AS00] Rakesh Agrawal and Ramakrishnan Srikant. "Privacy-preserving data mining." English. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00)*. Dellas, TX, USA: ACM New York, May 2000, pp. 439–450. ISBN: 1-58113-217-4. DOI: 10.1145/342009.335438. URL: http://dl.acm.org/citation.cfm?doid=342009.335438 (visited on 06/11/2015).

[Ate+07] Giuseppe Ateniese et al. "Provable data possession at untrusted stores." English. In: *Proceedings of the 14th ACM conference on Computer and communications security (CCS '07)*. Alexandria, VA, USA: ACM Press, Oct. 2007, pp. 598–609. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315318. URL: http://dl.acm.org/citation.cfm?id=1315245.1315318 (visited on 06/11/2015).

[Ayd+13] Erman Ayday et al. "Privacy-preserving processing of raw genomic data." English. In: *Proceedings 8th International Workshop on Data Privacy Management and 6th International WOrkshop on Autonomous Spontaneous Security (DPM 2013 and SETOP 2013)*. Vol. 8247. LNCS. Egham, UK: Springer Berlin Heidelberg, Sept. 2013, pp. 133–147. ISBN: 978-3-642-54568-9. DOI: 10.1007/978-3-642-54568-9_9. URL: http://link.springer.com/chapter/10.1007/978-3-642-54568-9_9 (visited on 06/11/2015).

[BA08] David Benjamin and Mikhail J. Atallah. "Private and Cheating-Free Outsourcing of Algebraic Computations." English. In: *Proceedings 6th Annual Conference on Privacy, Security and Trust (PST '08)*. Fredericton, NB, USA: IEEE, Oct. 2008, pp. 240–245. ISBN: 978-0-7695-3390-2. DOI: 10.1109/PST.2008.12. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4641291 (visited on 06/11/2015).

[BA10] Marina Blanton and Mehrdad Aliasgari. "Secure outsourcing of DNA searching via finite automata." English. In: *Proceedings 24th Annual IFIP WG 11.3 Working Conference*. Vol. 6166. LNCS. Rome, Italy: Springer Berlin Heidelberg, June 2010, pp. 49–64. ISBN: 978-3-642-13739-6. DOI: 10.1007/978-3-642-13739-6_4. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-13739-6_4 (visited on 06/11/2015).

[Bal+11] Pierre Baldi et al. "Countering GATTACA: efficient and secure testing of fully-sequenced human genomes." English. In: *Proceedings of the 18th ACM conference on Computer and communications security (CCS 2011)*. CCS '11. Chicago, IL, USA: ACM, 2011, pp. 691–702. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046785. URL: http://dl.acm.org/citation.cfm?doid=2046707.2046785 (visited on 05/27/2015).

[Bar+04] Z. Bar-Yossef et al. "Approximating edit distance efficiently." In: *45th Annual IEEE Symposium on Foundations of Computer Science* (2004). ISSN: 0-7695-2228-9. DOI: 10.1109/FOCS.2004.14.

[Bar+12] Elaine Barker et al. *Recommendation for Key Management Part 1 - Special Publication 800-57*. Tech. rep. 2012, pp. 1–147.

[Bar11] Thomas Jr. Barnett. *The Dawn of the Zettabyte Era*. English. June 2011. URL: http://blogs.cisco.com/news/the-dawn-of-the-zettabyte-era-infographic/ (visited on 06/11/2015).

[BB07] Mihir Bellare and Alexandra Boldyreva. "Deterministic and Efficiently Searchable Encryption." English. In: *Proceedings of the 27th annual international cryptology conference on Advances in cryptology (CRYPTO 2007)*. Vol. 4622. LNCS. Santa Barbara, CA, USA:

Springer Berlin Heidelberg, Aug. 2007, pp. 535–552. ISBN: 978-3-540-74143-5. DOI: 10.1007/978-3-540-74143-5_30. URL: http://link.springer.com/chapter/10.1007%2F978-3-540-74143-5_30 (visited on 04/28/2015).

[BB13] Peter T. Breuer and Jonathan P. Bowen. "A Fully Homomorphic Crypto-Processor Design." English. In: *Proceedings 5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013)*. Ed. by Jan Jürjens, Benjamin Livshits, and Riccardo Scandariato. Vol. 7781. LNCS. Paris, France: Springer Berlin Heidelberg, Feb. 2013, pp. 123–138. ISBN: 978-3-642-36562-1. DOI: 10.1007/978-3-642-36563-8_9. URL: http://link.springer.com/chapter/10.1007/978-3-642-36563-8_9 (visited on 05/07/2015).

[BBL12] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti. ""Better Than Nothing" Privacy with Bloom Filters: To What Extent?" In: *Proceedings of International Conference on Privacy in Statistical Databases*. Ed. by Josep Domingo-Ferrer and Ilenia Tinnirello. Palermo, Italy: Springer Berlin Heidelberg, Sept. 2012, pp. 348–363. ISBN: 978-3-642-33627-0. DOI: 10.1007/978-3-642-33627-0_27. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-33627-0_27 (visited on 06/09/2015).

[BC04] Steven M. Bellovin and William R Cheswick. *Privacy-Enhanced Searches Using Encrypted Bloom Filters*. English. Tech. rep. 2004/022. AT&T Research, Feb. 2004, pp. 1–12. URL: http://eprint.iacr.org/2004/022 (visited on 06/09/2015).

[BC14] Alexandra Boldyreva and Nathan Chenette. *Efficient Fuzzy Search on Encrypted Data*. Tech. rep. 2014/235. Georgia Institute of Technology, Mar. 2014, pp. 1–33. URL: https://eprint.iacr.org/2014/235 (visited on 04/28/2015).

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum disclosure proofs of knowledge." English. In: *Journal of Computer and System Sciences* 37.2 (Oct. 1988), pp. 156–189. DOI: 10.1016/0022-0000(88)90005-0. URL: http://www.sciencedirect.com/science/article/pii/0022000088900050 (visited on 06/11/2015).

[BCG11] Carlo Blundo, Emiliano De Cristofaro, and Paolo Gasti. "EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity." In: *Journal of Computer Security* 22.3 (2011), pp. 355–381. ISSN: 0926-227X. URL: http://dl.acm.org/citation.cfm?id=2597911 (visited on 04/24/2015).

[BD91] Thomas Beth and Yvo Desmedt. "Identification Tokens — or: Solving The Chess Grandmaster Problem." In: *Advances in Cryptology-CRYPT0' 90*. Springer Berlin Heidelberg, 1991, pp. 169–176. ISBN:

978-0-203-00365-7. DOI: 10.1007/3-540-38424-3_12. URL: http://link.springer.com/chapter/10.1007%2F3-540-38424-3_12.

[Bea+13] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers*. Tech. rep. June 2013. URL: https://eprint.iacr.org/2013/404.

[Bec15] Martin Beck. "Randomized Decryption (RD) Mode of Operation for Homomorphic Cryptography - Increasing Encryption, Communication and Storage Efficiency." English. In: *The Third International Workshop on Security and Privacy in Big Data (BigSecurity 2015)*. Hong Kong, China: IEEE, Apr. 2015, pp. 220–226. DOI: 10.1109/INFCOMW.2015.7179388. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7179388 (visited on 10/21/2015).

[Ben94] Josh Benaloh. "Dense probabilistic encryption." English. In: *Proceedings of the Workshop on Selected Areas of Cryptography*. 1994, pp. 1–26. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=68478 (visited on 06/11/2015).

[BF03] Dan Boneh and Matthew Franklin. "Identity-Based Encryption from the Weil Pairing." In: *SIAM Journal on Computing* 32 (2003), pp. 586–615. ISSN: 1095-7111. DOI: 10.1137/S0097539701398521. URL: http://epubs.siam.org/doi/abs/10.1137/S0097539701398521 (visited on 05/05/2015).

[BG02] B. Barak and O. Goldreich. "Universal arguments and their applications." English. In: *Proceedings 17th IEEE Annual Conference on Computational Complexity*. IEEE Comput. Soc, 2002, pp. 194–203. ISBN: 0-7695-1468-5. DOI: 10.1109/CCC.2002.1004355. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1004355 (visited on 06/11/2015).

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. "Packed Ciphertexts in LWE-Based Homomorphic Encryption." English. In: *Proceedings of 16th International Conference on Practice and Theory in Public-Key Cryptography*. Nara, Japan: Springer Berlin Heidelberg, 2013, pp. 1–13. ISBN: 978-3-642-36362-7. DOI: 10.1007/978-3-642-36362-7_1. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-36362-7_1 (visited on 04/21/2015).

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF formulas on ciphertexts." English. In: *Theory of Cryptography Conference (TCC) '05*. Vol. 3378. Cambridge, MA, USA: Springer Berlin Heidelberg, 2005, pp. 325–341. DOI: 10.1007/978-3-540-30576-7_18. URL: http://www.springerlink.com/index/wtt5caxkr94laxkg.pdf (visited on 02/17/2016).

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption without Bootstrapping." In: *Proceeding ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2011, pp. 309–325. DOI: 10.1145/2090236.2090262. URL: http://eprint.iacr.org/2011/277.

[Bic+09] Patrik Bichsel et al. "Anonymous credentials on a standard java card." In: *Proceedings of the 16th ACM conference on Computer and communications security CCS 09* (2009), pp. 600–600. ISSN: 9781605588940. DOI: 10.1145/1653662.1653734. URL: http://portal.acm.org/citation.cfm?doid=1653662.1653734.

[Bil05] Philip Bille. "A survey on tree edit distance and related problems." English. In: *Theoretical Computer Science* 337.1-3 (June 2005), pp. 217–239. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2004.12.030. URL: http://www.sciencedirect.com/science/article/pii/S0304397505000174 (visited on 06/09/2015).

[BK13] Martin Beck and Florian Kerschbaum. "Approximate two-party privacy-preserving string matching with linear complexity." In: *Proceedings of the 2013 IEEE International Congress on Big Data (BigData Congress 2013)*. Santa Clara, CA: IEEE, Sept. 2013, pp. 31–37. ISBN: 9780768550060. DOI: 10.1109/BigData.Congress.2013.14. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6597116 (visited on 03/16/2015).

[BKD12] Elaine Barker, John Kelsey, and Computer Security Division. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. Tech. rep. 2012.

[Bla+12a] Marina Blanton et al. "Secure and Efficient Outsourcing of Sequence Comparisons." In: *Computer Security–ESORICS* 7459 (2012), pp. 505–522. DOI: 10.1007/978-3-642-33167-1_29. URL: http://www.springerlink.com/index/Q473446220270882.pdf.

[Bla+12b] EO Blass et al. "PRISM–Privacy-Preserving Search in MapReduce." In: *Privacy Enhancing Technologies* (2012). URL: http://www.springerlink.com/index/0720TJP211W4Q6U7.pdf.

[Blu+91] M. Blum et al. "Checking the correctness of memories." In: *Proceedings 32nd Annual Symposium of Foundations of Computer Science*. IEEE Comput. Soc. Press, Sept. 1991, pp. 90–99. ISBN: 0-8186-2445-0. DOI: 10.1109/SFCS.1991.185352. URL: http://dl.acm.org/citation.cfm?id=123229.123261.

[BM93] Josh Benaloh and Michael de Mare. "One-way accumulators: a decentralized alternative to digital signatures." In: *Proceeding EUROCRYPT '93 Workshop on the theory and application of cryptographic techniques on Advances in cryptology*. Lofthus, Norwa: Springer-

Verlag New York, Inc., May 1993, pp. 274–285. ISBN: 978-3-540-48285-7. DOI: 10.1007/3-540-48285-7_24. URL: http://link.springer.com/chapter/10.1007%2F3-540-48285-7_24 (visited on 05/27/2015).

[BMS07] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. "Scaling up all pairs similarity search." In: *Proceedings of the 16th international conference on World Wide Web - WWW '07*. ACM Press, May 2007, pp. 131–131. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242591. URL: http://dl.acm.org/citation.cfm?id=1242572.1242591 (visited on 06/11/2015).

[Bon+07] Dan Boneh et al. "Public key encryption that allows PIR queries." In: (Aug. 2007), pp. 50–67. ISSN: 3-540-74142-9, 978-3-540-74142-8. URL: http://dl.acm.org/citation.cfm?id=1777777.1777783.

[Bor+12] Julia Borghoff et al. "PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications." In: *Advances in Cryptology – ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-34960-7. URL: http://www.springerlink.com/index/10.1007/978-3-642-34961-4.

[Bou85] J. Bourgain. "On lipschitz embedding of finite metric spaces in Hilbert space." In: *Israel Journal of Mathematics* 52.1-2 (Mar. 1985), pp. 46–52. DOI: 10.1007/BF02776078. URL: http://www.springerlink.com/index/10.1007/BF02776078.

[BP10] Joan Boyar and René Peralta. "A new combinational logic minimization technique with applications to cryptology." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6049 LNCS.2 (2010), pp. 178–189. ISSN: 3642131921. DOI: 10.1007/978-3-642-13193-6_16.

[Bra12] Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP." In: *Advances in Cryptology – CRYPTO 2012*. 2012, pp. 868–886. DOI: 10.1007/978-3-642-32009-5_50. URL: http://eprint.iacr.org/2012/078.

[Bro97] A.Z. Broder. "On the resemblance and containment of documents." In: *Proceedings. Compression and Complexity of SEQUENCES 1997*. (Cat. No.97TB100171). Salerno: IEEE Comput. Soc, June 1997, pp. 21–29. ISBN: 0-8186-8132-2. DOI: 10.1109/SEQUEN.1997.666900. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=666900 (visited on 05/27/2015).

[BS10] Tobias Bachteler and Rainer Schnell. "An empirical comparison of approaches to approximate string matching in private record linkage." In: *Proceedings of Statistics Canada* (2010). URL: http://www.uni-due.de/~hq0215/documents/Draft_Ottawa_Comparison.pdf.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. "Functional encryption: definitions and challenges." In: (Mar. 2011), pp. 253–273. ISSN: 978-3-642-19570-9. URL: http://dl.acm.org/citation.cfm?id=1987260.1987281.

[Bul+00] Ahto Buldas et al. "Accountable certificate management using undeniable attestations." In: *Proceedings of the 7th ACM conference on Computer and communications security - CCS '00* (2000), pp. 9–17. ISSN: 1581132034. DOI: 10.1145/352600.352604. URL: http://portal.acm.org/citation.cfm?doid=352600.352604 (visited on 04/17/2015).

[BW07] Dan Boneh and Brent Waters. "Conjunctive, Subset, and Range Queries on Encrypted Data." In: *In proceedings of TCC'07, LNCS 4392* (2007), pp. 535–554.

[Cal+07] J. Callas et al. *OpenPGP Message Format*. RFC 4880 (Proposed Standard). Updated by RFC 5581. Internet Engineering Task Force, Nov. 2007. URL: http://www.ietf.org/rfc/rfc4880.txt.

[Can+15] Anne Canteaut et al. *How to Compress Homomorphic Ciphertexts*. Tech. rep. 2015. URL: http://eprint.iacr.org/2015/113 (visited on 05/27/2015).

[Can06] Christophe De Cannière. "TRIVIUM: A Stream Cipher Construction Inspired by Block Cipher Design Principles." In: *9th International Conference, ISC 2006*. Vol. 507932. Springer Berlin Heidelberg, 2006, pp. 171–186. ISBN: 978-3-540-38343-7. DOI: 10.1007/11836810_13. URL: http://link.springer.com/chapter/10.1007%2F11836810_13.

[Cao+11] Ning Cao et al. "Privacy-preserving multi-keyword ranked search over encrypted cloud data." In: *INFOCOM, 2011 Proceedings IEEE*. Shanghai, China, Apr. 2011, pp. 829–837. ISBN: 978-1-4244-9919-9. DOI: 10.1109/INFCOM.2011.5935306. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5935306 (visited on 04/28/2015).

[Cao+14] Ning Cao et al. "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data." In: 25.1 (2014), pp. 222–233. DOI: 10.1109/TPDS.2013.45. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6674958 (visited on 05/19/2015).

[Cas+13] David Cash et al. "Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries." In: *Advances in Cryptology – CRYPTO 2013*. Santa Barbara, CA, USA, Aug. 2013, pp. 353–373. ISBN: 978-3-642-40041-4. DOI: 10.1007/978-3-642-40041-4_20. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-40041-4_20 (visited on 04/28/2015).

[Cas+14] David Cash et al. "Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation." In: *Beiefings in NDSS Symposium 2014*. San Diego, California, USA, Feb. 2014, pp. 23–26. URL: http://www.internetsociety.org/doc/dynamic-searchable-encryption-very-large-databases-data-structures-and-implementation (visited on 04/28/2015).

[CD97] Ronald Cramer and Ivan B. Damgård. "Fast and Secure Immunization Against Adaptive Man-in-the-Middle Impersonation." In: *Proceedings International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '97)*. LNCS. Konstanz, Germany: Springer Berlin Heidelberg, May 1997, pp. 75–87. ISBN: 978-3-540-69053-5. DOI: 10.1007/3-540-69053-0_7. URL: http://link.springer.com/chapter/10.1007%2F3-540-69053-0_7 (visited on 06/11/2015).

[CDN01] Ronald Cramer, Ivan B. Damgård, and Jesper Buus Nielsen. "Multiparty Computation from Threshold Homomorphic Encryption." English. In: *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. Innsbruck, Austria: Springer Berlin Heidelberg, May 2001, pp. 280–300. ISBN: 978-3-540-44987-4. DOI: 10.1007/3-540-44987-6_18. URL: http://link.springer.com/chapter/10.1007%2F3-540-44987-6_18 (visited on 09/30/2015).

[CGT11] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. *Fast and Private Computation of Cardinality of Set Intersection and Union*. Tech. rep. 2011, pp. 1–19. URL: http://eprint.iacr.org/2011/141.

[Cha83] David Chaum. "Blind Signatures for Untraceable Payments." In: *Advances in Cryptology - Proceedings of Crypto 82*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Springer US, 1983, pp. 199–203. ISBN: 978-1-4757-0604-8. DOI: 10.1007/978-1-4757-0602-4. URL: http://link.springer.com/10.1007/978-1-4757-0602-4.

[Che+12] Yangyi Chen et al. "Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds." In: *NDSS 2012, The Internet Society*. 2012. URL: http://www.internetsociety.org/large-scale-privacy-preserving-mapping-human-genomic-sequences-hybrid-clouds.

[Che+13] Jung Hee Cheon et al. "Batch Fully Homomorphic Encryption over the Integers." In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-38347-2. URL: http://link.springer.com/10.1007/978-3-642-38348-9.

[CK13] JH Cheon and Jinsu Kim. "An Approach to Reduce Storage for Homomorphic Computations." In: *IACR Cryptology ePrint Archive* (2013). URL: eprint.iacr.org/2013/710 (visited on 03/11/2015).

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "Practical Multilinear Maps over the Integers." In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 1–25. ISBN: 978-3-642-40040-7. DOI: 10.1007/978-3-642-40041-4. URL: http://link.springer.com/10.1007/978-3-642-40041-4.

[CM05] Yan-Cheng Chang and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." In: *Proceeding ACNS'05 Proceedings of the Third international conference on Applied Cryptography and Network Security*. Ed. by John Ioannidis, Angelos Keromytis, and Moti Yung. Vol. 3531. Lecture Notes in Computer Science. Springer Berlin Heidelberg, June 2005, pp. 442–455. ISBN: 978-3-540-26223-7. DOI: 10.1007/11496137_30. URL: http://dl.acm.org/citation.cfm?id=2134532.2134562 (visited on 04/28/2015).

[CM99] Jan Camenisch and Markus Michels. "Proving in zero-knowledge that a number is the product of two safe primes." In: *Advances in Cryptology - EUROCRYPT'99* November (1999), pp. 107–122. DOI: 10.1007/3-540-48910-X_8. URL: http://link.springer.com/chapter/10.1007/3-540-48910-X_8 (visited on 03/11/2015).

[Coc01] Clifford Cocks. "An identity based encryption scheme based on quadratic residues." In: *Cryptography and Coding* 2260 (2001), pp. 360–363. ISSN: 978-3-540-43026-1. DOI: 10.1007/3-540-45325-3_32. URL: http://link.springer.com/chapter/10.1007/3-540-45325-3_32.

[CRJ10] Ken Christensen, Allen Roginsky, and Miguel Jimeno. "A new analysis of the false positive rate of a Bloom filter." In: *Information Processing Letters* 110.21 (Oct. 2010), pp. 944–949. DOI: 10.1016/j.ipl.2010.07.024. URL: http://www.sciencedirect.com/science/article/pii/S0020019010002425 (visited on 06/09/2015).

[CS94] Ronald Cramer and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols." English. In: *Proceedings 14th Annual International Cryptology Conference (CRYPTO '94)*. Vol. 839. LNCS. Santa Barbara, CA, USA: Springer Berlin Heidelberg, 1994, pp. 174–187. ISBN: 978-3-540-48658-9. DOI: 10.1007/3-540-48658-5_19. URL: http://link.springer.com/chapter/10.1007%2F3-540-48658-5_19 (visited on 06/11/2015).

[CT10] Emiliano De Cristofaro and Gene Tsudik. "Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity." English. In: *Financial Cryptography and Data Security - Lecture Notes in Computer Science*. Vol. 6052/2010. LNCS. Tenerife, Canary Islands: Springer Berlin Heidelberg, 2010, pp. 143–159. ISBN: 978-3-642-14577-3. DOI: 10.1007/978-3-642-14577-3_13. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-14577-3_13 (visited on 04/21/2015).

[CT12] Emiliano De Cristofaro and Gene Tsudik. "On the performance of certain Private Set Intersection protocols." In: *Cryptology ePrint Archive, Report 2012/054* Trust (Feb. 2012), pp. 1–20. URL: http://eprint.iacr.org/2012/054.

[Cur+06] Reza Curtmola et al. "Searchable symmetric encryption: Improved Definitions and Efficient Constructions." English. In: *Proceedings of the 13th ACM conference on Computer and communications security (CCS '06)*. ACM Press, Oct. 2006, pp. 79–79. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180417. URL: http://dl.acm.org/citation.cfm?id=1180405.1180417 (visited on 04/28/2015).

[CZ09] Jan Camenisch and Gregory M. Zaverucha. "Private Intersection of Certified Sets." In: *Proceedings 13th International Conference on Financial Cryptography and Data Security (FC 2009)*. LNCS. Accra Beach, Barbados: Springer Berlin Heidelberg, 2009, pp. 108–127. ISBN: 978-3-642-03549-4. DOI: 10.1007/978-3-642-03549-4_7. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-03549-4_7 (visited on 06/11/2015).

[DA01] Wenliang Du and Mikhail J. Atallah. "Protocols for Secure Remote Database Access with Approximate Matching." In: *E-Commerce Security and Privacy*. Advances in Information Security 2. Springer US, 2001, pp. 87–111. ISBN: 978-1-4615-1467-1. URL: http://link.springer.com/chapter/10.1007/978-1-4615-1467-1_6 (visited on 04/21/2015).

[Dac+09] Dana Dachman-Soled et al. "Efficient robust private set intersection." In: *International Journal of Applied Cryptography*. Vol. 2. Paris: Springer Berlin Heidelberg, June 2009, pp. 125–142. ISBN: 978-3-642-01957-9. DOI: 10.1007/978-3-642-01957-9_8. URL: http:

//link.springer.com/chapter/10.1007%2F978-3-642-01957-9_8 (visited on 03/11/2015).

[Dan+04] Dan Boneh et al. "Public Key Encryption with Keyword Search." English. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Christian Cachin and Jan L. Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer Berlin Heidelberg, May 2004, pp. 506–522. ISBN: 978-3-540-21935-4. DOI: 10.1007/978-3-540-24676-3_30. URL: http://link.springer.com/chapter/10.1007%2F978-3-540-24676-3_30 (visited on 05/27/2015).

[DDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. "KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5747 LNCS (2009), pp. 272–288. ISSN: 364204137X. DOI: 10.1007/978-3-642-04138-9_20.

[De 14] Emiliano De Cristofaro. "Genomic Privacy and the Rise of a New Research Community." In: *IEEE Security & Privacy* 12.2 (Mar. 2014), pp. 80–83. DOI: 10.1109/MSP.2014.24. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6798562.

[Dev86] Luc Devroye. *Non-Uniform Random Variate Generation*. New York: Springer New York, 1986. ISBN: 978-1-4613-8643-8. URL: http://link.springer.com/book/10.1007%2F978-1-4613-8643-8 (visited on 03/11/2015).

[DFM98] GI Davida, Yair Frankel, and BJ Matt. "On enabling secure applications through off-line biometric identification." In: *... . 1998 IEEE Symposium on* (1998), pp. 148–157. ISSN: 0818683864. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=674831.

[DHS14] Y Doroz, Yin Hu, and Berk Sunar. *Homomorphic AES Evaluation Using NTRU*. Tech. rep. 2014, pp. 1–16. URL: https://eprint.iacr.org/2014/039.pdf.

[DJ01] Ivan B. Damgård and Mads J. Jurik. "A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System." In: *Proceeding PKC '01 Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*. Cheju Island, Korea: Springer Berlin Heidelberg, Feb. 2001, pp. 119–136. ISBN: 978-3-540-44586-9. DOI: 10.1007/3-540-44586-2_9. URL: http://link.springer.com/chapter/10.1007%2F3-540-44586-2_9 (visited on 05/07/2015).

[Dom08] Josep Domingo-Ferrer. "A Survey of Inference Control Methods for Privacy-Preserving Data Mining." In: *PrivacyPreserving Data Mining*. Vol. 34. Springer US, 2008, pp. 53–80. ISBN: 978-0-387-70992-5. URL: http://link.springer.com/chapter/10.1007%2F978-0-387-70992-5_3 (visited on 06/09/2015).

[Don+13] Q Dong et al. "Fuzzy keyword search over encrypted data in the public key setting." English. In: *Proceedings 14th International Conference on Web-Age Information Management (WAIM 2013)*. Vol. 7923. LNCS. Beidaihe, China: Springer Berlin Heidelberg, 2013, pp. 729–740. ISBN: 978-3-642-38562-9. DOI: 10.1007/978-3-642-38562-9_74. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-38562-9_74 (visited on 04/28/2015).

[Dur+12] Elizabeth Ashley Durham et al. "Quantifying the Correctness, Computational Complexity, and Security of Privacy-Preserving String Comparators for Record Linkage." In: *An international journal on information fusion* 13.4 (Oct. 2012), pp. 245–259. DOI: 10.1016/j.inffus.2011.04.004. URL: http://www.ncbi.nlm.nih.gov/pubmed/22904698 (visited on 05/30/2015).

[Dur+14] Elizabeth Ashley Durham et al. "Composite Bloom Filters for Secure Record Linkage." In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (Dec. 2014), pp. 2956–2968. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.91. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6529084 (visited on 04/23/2015).

[Dwo06] Cynthia Dwork. "Differential privacy." In: *in International Colloquium on Automata, Languages and Programming (ICALP)*. Vol. 4052. Springer, 2006, pp. 1–12. ISBN: 978-3-540-35908-1. DOI: 10.1007/11787006_1. URL: http://link.springer.com/chapter/10.1007%2F11787006_1 (visited on 04/17/2015).

[Dwo08] Cynthia Dwork. "Differential privacy: a survey of results." In: *Proceedings of the 5th international conference on Theory and applications of models of computation*. TAMC'08. Springer-Verlag, 2008, pp. 1–19. ISBN: 3-540-79227-9 978-3-540-79227-7. URL: http://portal.acm.org/citation.cfm?id=1791834.1791836 (visited on 04/17/2015).

[EGS03] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. "Limiting privacy breaches in privacy preserving data mining." In: *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '03*. ACM Press, 2003, pp. 211–222. ISBN: 1-58113-670-6. DOI: 10.1145/773153.773174. URL: http://portal.acm.org/citation.cfm?doid=773153.773174.

[ElG84] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." In: *Advances in Cryptology*. Santa Barbara, California, USA: Springer Berlin Heidelberg, Aug. 1984, pp. 10–18. ISBN: 978-3-540-39568-3. DOI: 10.1007/3-540-39568-7_2. URL: http://link.springer.com/chapter/10.1007%2F3-540-39568-7_2 (visited on 05/04/2015).

[EOM14] Kaoutar Elkhiyaoui, Melek Onen, and Refik Molva. "Privacy preserving delegated word search in the cloud." In: *Proceedings of 11th International conference on Security and Cryptography (SECRYPT '14)*. Vienna, Austria, Aug. 2014. URL: http://www.eurecom.fr/publication/4345 (visited on 04/28/2015).

[Erw+09] Chris Erway et al. "Dynamic provable data possession." English. In: *Proceedings 16th ACM Conference on Computer and Communications Security (CCS 2009)*. Chicago, IL, USA: ACM New York, Nov. 2009, pp. 213–222. ISBN: 978-1-60558-894-0. DOI: 10.1145/1653662.1653688. URL: http://dl.acm.org/citation.cfm?doid=1653662.1653688 (visited on 06/11/2015).

[Fau+13] Simon Fau et al. "Towards Practical Program Execution over Fully Homomorphic Encryption Schemes." English. In: *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. IEEE, Oct. 2013, pp. 284–290. ISBN: 978-0-7695-5094-7. DOI: 10.1109/3PGCIC.2013.48. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6681241.

[FN94] Amos Fiat and Moni Naor. "Broadcast encryption." In: *Proceedings of the 13th annual international cryptology conference on Advances in cryptology* 773 (1994), pp. 480–491. ISSN: 0-387-57766-1. DOI: 10.1007/3-540-48329-2. URL: http://dl.acm.org/citation.cfm?id=188105.188198.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient private matching and set intersection." English. In: *Proceedings International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004)*. Vol. 3027. LNCS. Interlaken, Switzerland: Springer Berlin Heidelberg, May 2004, pp. 1–19. ISBN: 978-3-540-21935-4. DOI: 10.1007/978-3-540-24676-3_1. URL: http://link.springer.com/chapter/10.1007/978-3-540-24676-3_1 (visited on 06/09/2015).

[Fra+12] Martin Franz et al. "Towards Secure Bioinformatics Services." In: *Financial Cryptography and Data Security - Lecture Notes in Computer Science* 7035 (2012), pp. 276–283.

[Fre+05] Michael J Freedman et al. "Keyword Search and Oblivious Pseudorandom Functions Michael." English. In: *Proceedings Second Theory of Cryptography Conference (TCC 2005)*. Ed. by Joe Kilian. Vol. 3378. LNCS. Cambridge, MA, USA: Springer Berlin Heidelberg, Feb. 2005, pp. 303–324. ISBN: 978-3-540-24573-5. DOI: 10.1007/978-3-540-30576-7_17. URL: http://link.springer.com/chapter/10.1007/978-3-540-30576-7_17 (visited on 06/11/2015).

[FV12] Junfeng Fan and Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Tech. rep. 2012. URL: https://eprint.iacr.org/2012/144.

[GB08] Shafi Goldwasser and Mihir Bellare. *Lecture notes on cryptography*. English. Tech. rep. Cambridge, MA, USA: Massachusetts Institute of Technology (MIT), July 2008, pp. 1–289. URL: http://cseweb.ucsd.edu/~mihir/papers/gb.html (visited on 01/22/2016).

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices." In: *Proceedings of the 41st annual ACM symposium on Theory of computing*. STOC '09. ACM, 2009, pp. 169–178. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536440. URL: http://doi.acm.org/10.1145/1536414.1536440.

[GGM00] Irene Gassko, Peter S. Gemmell, and Philip MacKenzie. "Efficient and fresh certification." English. In: *Proceedings Third International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*. Vol. 1751. LNCS. Melbourne, Victoria, Australia: Springer Berlin Heidelberg, Jan. 2000, pp. 342–353. ISBN: 978-3-540-46588-1. DOI: 10.1007/978-3-540-46588-1_23. URL: http://link.springer.com/chapter/10.1007%2F978-3-540-46588-1_23 (visited on 06/11/2015).

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. *Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-14622-0. URL: http://link.springer.com/chapter/10.1007/978-3-642-14623-7_25.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. "Fully Homomorphic Encryption with Polylog Overhead." English. In: *Proceedings 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)*. Vol. 7237. LNCS. Cambridge, UK: Springer Berlin Heidelberg, Apr. 2012, pp. 465–482. ISBN: 978-3-642-29011-4. DOI: 10.1007/978-3-642-29011-4_28. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-29011-4_28 (visited on 06/11/2015).

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. "Homomorphic Evaluation of the AES Circuit." English. In: *Advances in Cryptology – CRYPTO 2012 Lecture Notes in Computer Science*. Vol. 7417. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Aug. 2012, pp. 850–867. ISBN: 978-3-642-32009-5. DOI: 10.1007/978-3-642-32009-5_49. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-32009-5_49 (visited on 06/10/2015).

[GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. "Similarity Search in High Dimensions via Hashing." In: *VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases*. Ed. by Malcolm P. Atkinson et al. Morgan Kaufmann Publishers Inc., Sept. 1999, pp. 518–529. ISBN: 1-55860-615-7. URL: http://dl.acm.org/citation.cfm?id=645925.671516 (visited on 04/21/2015).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption & how to play mental poker keeping secret all partial information." In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing - STOC '82*. ACM Press, May 1982, pp. 365–377. ISBN: 0-89791-070-2. DOI: 10.1145/800070.802212. URL: http://dl.acm.org/citation.cfm?id=800070.802212.

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption." In: *Journal of Computer and System Sciences* 28.2 (Apr. 1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9. URL: http://linkinghub.elsevier.com/retrieve/pii/0022000084900709.

[GN12] C Gehrmann and M Naslund. *ECRYPT II Yearly Report on Algorithms and Keysizes*. Tech. rep. Sept. 2012. URL: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:yearly+report+on+algorithms+and+keysizes#1.

[Goh04] Eu-jin Goh. *Secure Indexes*. English. Tech. rep. Stanford, CA, USA: Stanford University, 2004, pp. 1–19. URL: http://eprint.iacr.org/2003/216 (visited on 05/27/2015).

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume II Basic Applications*. 2004. ISBN: 978-0-521-11991-7.

[Gol87] O. Goldreich. "Towards a theory of software protection and simulation by oblivious RAMs." English. In: *Proceedings of the 19th Annual ACM conference on Theory of computing (STOC '87)*. ACM Press, Jan. 1987, pp. 182–194. ISBN: 0-89791-221-7. DOI: 10.1145/28395.28416. URL: http://dl.acm.org/citation.cfm?id=28395.28416 (visited on 06/11/2015).

[Goo09] Michael T. Goodrich. "The Mastermind Attack on Genomic Data." In: *2009 30th IEEE Symposium on Security and Privacy*. IEEE, May 2009, pp. 204–218. ISBN: 978-0-7695-3633-0. DOI: 10.1109/SP.2009.4. URL: http://dl.acm.org/citation.cfm?id=1607723.1608134.

[Goy+06] Vipul Goyal et al. "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data." English. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. ACM New York, 2006, pp. 89–98. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180418. URL: http://dl.acm.org/citation.cfm?doid=1180405.1180418 (visited on 06/09/2015).

[GR12] John Gantz and David Reinsel. *The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East*. Tech. rep. Dec. 2012, pp. 16–16. URL: http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.

[Gra+01] Luis Gravano et al. "Approximate String Joins in a Database (Almost) for Free." In: *Proceedings of the 27th International Conference on Very Large Data Bases* (Sept. 2001), pp. 491–500. ISSN: 1-55860-804-4. URL: http://dl.acm.org/citation.cfm?id=645927.672200.

[Gro10] Jens Groth. "A verifiable secret shuffle of homomorphic encryptions." English. In: *Journal of Cryptology* 23 (2010), pp. 546–579. ISSN: 1432-1378. DOI: 10.1007/s00145-010-9067-9. URL: http://link.springer.com/article/10.1007%2Fs00145-010-9067-9 (visited on 04/23/2015).

[GS02] Craig Gentry and Alice Silverberg. *Hierachical ID-Based Cryptography*. Vol. 6477. 2002. ISBN: 978-3-642-17372-1. URL: http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2010.html#JagerKSS10.

[GSW04] Philippe Golle, Jessica Staddon, and Brent Waters. "Secure Conjunctive Keyword Search Over Encrypted Data." In: *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security* (2004), pp. 31–45.

[GTS01] M.T. Goodrich, R. Tamassia, and A. Schwerin. "Implementation of an authenticated dictionary with skip lists and commutative hashing." In: *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01* 2 (2001), pp. 68–82. ISSN: 0-7695-1212-7. DOI: 10.1109/DISCEX.2001.932160.

[Guo+11] Jian Guo et al. "The LED Block Cipher." In: *Proceedings of CHES 2011* (2011). URL: http://eprint.iacr.org/2012/600.

[Ham50] Richard Wesley Hamming. "Error-Detecting and Error-Correcting Codes." In: *Bell System Technical Journal* 29 (1950), pp. 147–160.

[HD80] Patrick A. V. Hall and Geoff R. Dowling. "Approximate String Matching." In: *ACM Computing Surveys* 12.4 (Dec. 1980), pp. 381–402. DOI: 10.1145/356827.356830. URL: http://dl.acm.org/citation.cfm?id=356827.356830.

[HEK11] Yan Huang, David Evans, and Jonathan Katz. "Faster secure two-party computation using garbled circuits." In: *USENIX Security Symposium* (2011). URL: http://www.usenix.org/event/sec11/tech/full_papers/Huang.pdf.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. "Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?" In: *NDSS* (2012). URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.220.9698.

[HF11] Rob Hall and Stephen E. Fienberg. "Privacy Preserving Record Linkage." In: *Privacy in Statistical Databases - Lecture Notes in Computer Science* 6344/2011 (2011), pp. 269–283. DOI: 10.1007/978-3-642-15838-4_24. URL: http://www.springerlink.com/content/m32xq75577623387/.

[HL02] Jeremy Horwitz and Ben Lynn. "Toward Hierarchical Identity-Based Encryption." In: *Advances in CryptologyEUROCRYPT 2002*. Vol. 2332. 2002, pp. 466–481. ISBN: 978-3-540-43553-2. DOI: 10.1007/3-540-46035-7_31. URL: http://www.springerlink.com/index/8QA6Q2VN0E5N6DYQ.pdf.

[HL08] Carmit Hazay and Yehuda Lindell. "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries." In: *TCC'08 Proceedings of the 5th conference on Theory of cryptography* (Mar. 2008), pp. 155–175. ISSN: 3-540-78523-X, 978-3-540-78523-1. URL: http://dl.acm.org/citation.cfm?id=1802614.1802628.

[HN10] Carmit Hazay and Kobbi Nissim. "Efficient Set Operations in the Presence of Malicious Adversaries." English. In: *Proceedings 13th International Conference on Practice and Theory in Public Key Cryptography (PKC 2010)*. Vol. 6056. LNCS. Paris, France: Springer Berlin Heidelberg, May 2010, pp. 312–331. ISBN: 978-3-642-13013-7. DOI: 10.1007/978-3-642-13013-7_19. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-13013-7_19 (visited on 06/09/2015).

[HPN11] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. "Differential privacy under fire." In: *SEC'11 Proceedings of the 20th USENIX conference on Security*. Aug. 2011, pp. 33–33. URL: http://dl.acm.org/citation.cfm?id=2028067.2028100 (visited on 04/17/2015).

[Hsi08] Paul Hsieh. *Misconceptions about rand()*. 2008. URL: http://www.azillionmonkeys.com/qed/random.html (visited on 09/30/2015).

[Hu13] Yin Hu. "Improving the Efficiency of Homomorphic Encryption Schemes." PhD thesis. May 2013.

[Ibr+13] a. Ibrahim et al. "Towards Efficient Yet Privacy-Preserving Approximate Search in Cloud Computing." In: *The Computer Journal* 57.2 (May 2013), pp. 241–254. DOI: 10.1093/comjnl/bxt045. URL: http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/bxt045 (visited on 04/28/2015).

[Ide15] Identity Theft Resource Center. *Data Breaches*. English. Sept. 2015. URL: http://www.idtheftcenter.org/id-theft/data-breaches.html (visited on 09/30/2015).

[IG06] Max Ingman and U Gyllensten. "mtDB: Human Mitochondrial Genome Database, a resource for population genetics and medical sciences." In: *Nucleic Acids Research* 34 (2006), pp. 749–751.

[IKK12] MS Islam, Mehmet Kuzu, and Murat Kantarcioglu. "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation." In: *Briefings of NDSS Symposium 2012*. San Diego, California, USA, Feb. 2012, p. 15. URL: http://www.internetsociety.org/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation (visited on 04/17/2015).

[IM04] Piotr Indyk and Jiri Matousek. "Low-Distortion Embeddings of Finite Metric Spaces." In: *Handbook of Discrete and Computational Geometry* (2004), pp. 177–196. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.2484.

[IM98] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98)*. ACM Press, May 1998, pp. 604–613. ISBN: 0-89791-962-9. DOI: 10.1145/276698.276876. URL: http://dl.acm.org/citation.cfm?id=276698.276876 (visited on 06/11/2015).

[Ish+04] Yuval Ishai et al. "Batch codes and their applications." In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing - STOC '04*. ACM Press, June 2004, pp. 262–262. ISBN: 1-58113-852-0. DOI: 10.1145/1007352.1007396. URL: http://dl.acm.org/citation.cfm?id=1007352.1007396.

[Ish+06] Yuval Ishai et al. "Cryptography from Anonymity." In: *Proceedings 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06)*. Berkeley, CA, USA: IEEE, Oct. 2006, pp. 239–248. ISBN: 0-7695-2720-5. DOI: 10.1109/FOCS.2006.25. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031360 (visited on 06/11/2015).

[Jac01] Paul Jaccard. "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines." In: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37 (1901), pp. 241–272. URL: http://www.researchgate.net/profile/Paul_Jaccard/publication/243457811_Distribution_de_la_flore_alpine_dans_le_bassin_des_Dranses_et_dans_quelques_rgions_voisines/links/02e7e51d0a6c7c7705000000.

[JK07] Ari Juels and Burton S. Kaliski. "PoRs: proofs of retrievability for large files." In: *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*. ACM Press, Oct. 2007, pp. 584–584. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315317. URL: http://dl.acm.org/citation.cfm?id=1315245.1315317 (visited on 04/21/2015).

[JKS08] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. "Towards Practical Privacy for Genomic Computation." English. In: *Proceedings IEEE Symposium on Security and Privacy (SP 2008)*. Oakland, CA, USA: IEEE, May 2008, pp. 216–230. ISBN: 978-0-7695-3168-7. DOI: 10.1109/SP.2008.34. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4531155 (visited on 04/21/2015).

[JL10] Stanislaw Jarecki and Xiaomin Liu. "Fast Secure Computation of Set Intersection." In: *Security and Cryptography for Networks*. Ed. by Juan A. Garay and Roberto De Prisco. Springer Berlin Heidelberg, Sept. 2010, pp. 418–435. ISBN: 10.1007/978-3-642-15317-4_26.

[JL13] Marc Joye and Benoît Libert. "Efficient Cryptosystems from 2k-th Power Residue Symbols." English. In: *Proceedings 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2013)*. Vol. 7881. LNCS. Athens, Greece: Springer Berlin Heidelberg, 2013, pp. 76–92. ISBN: 978-3-642-38348-9. DOI: 10.1007/978-3-642-38348-9_5. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-38348-9_5 (visited on 06/11/2015).

[JW99] Ari Juels and Martin Wattenberg. "A fuzzy commitment scheme." In: *Proceedings of the 6th ACM conference on Computer and communications security - CCS '99*. ACM Press, Nov. 1999, pp. 28–36. ISBN: 1-58113-148-8. DOI: 10.1145/319709.319714. URL: http:

//dl.acm.org/citation.cfm?id=319709.319714 (visited on 04/23/2015).

[Kal98] B. Kaliski. *PKCS #7: Cryptographic Message Syntax Version 1.5*. RFC 2315 (Informational). Internet Engineering Task Force, Mar. 1998. URL: http://www.ietf.org/rfc/rfc2315.txt.

[Kan+08] Murat Kantarcioglu et al. "A cryptographic approach to securely share and query genomic sequences." In: *IEEE Transactions on Information Technology in Biomedicine* 12.5 (2008), pp. 606–617. ISSN: 1558-0032. DOI: 10.1109/TITB.2007.908465. URL: http://www.ncbi.nlm.nih.gov/pubmed/18779075 (visited on 03/11/2015).

[KBS14] Florian Kerschbaum, Martin Beck, and Dagmar Schönfeld. "Inference Control for Privacy-Preserving Genome Matching." In: Amsterdam, Netherlands, July 2014. URL: http://arxiv.org/abs/1405.0205 (visited on 03/16/2015).

[Ker11] Florian Kerschbaum. "Secure conjunctive keyword searches for unstructured text." In: *2011 5th International Conference on Network and System Security*. Milan, Italy: IEEE, Sept. 2011, pp. 285–289. ISBN: 978-1-4577-0458-1. DOI: 10.1109/ICNSS.2011.6060016. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6060016 (visited on 04/28/2015).

[Ker12] Florian Kerschbaum. "Collusion-Resistant Outsourcing of Private Set Intersection." In: *SAC '12 Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 2012, pp. 1451–1456. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2232008. URL: http://dl.acm.org/citation.cfm?id=2232008 (visited on 04/21/2015).

[Kha05] Dang Tran Khanh. "Privacy-Preserving Basic Operations on Outsourced Search Trees." English. In: *Proceedings 21st International Conference on Data Engineering Workshops (ICDEW '05)*. Tokyo, Japan: IEEE, Apr. 2005, pp. 1194–1201. ISBN: 0-7695-2657-8. DOI: 10.1109/ICDE.2005.264. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=1647807&contentType=Conference+Publications (visited on 06/11/2015).

[Kik+99] H Kikuchi et al. "Performance Evaluation of Certicate Revocation Using k-Valued Hash Tree." In: *Proc. ISW'99* 1729 (1999), pp. 103–117.

[Kir10] Marshall Kirkpatrick. *Google CEO Schmidt: "People Aren't Ready for the Technology Revolution"*. English. Aug. 2010. URL: http://readwrite.com/2010/08/04/google_ceo_schmidt_people_arent_ready_for_the_tech (visited on 09/30/2015).

[KK08] Murat Kantarcioglu and Onur Kardes. "Privacy-preserving data mining in the malicious model." In: *International Journal of Information and Computer Security* 2.4 (2008), pp. 353–353. DOI: 10.1504/IJICS.2008.022488. URL: http://www.inderscience.com/link.php?id=22488.

[KLG13] Florian Kerschbaum, Hoon Wei Lim, and Ivan Gudymenko. "Privacy-preserving Billing for e-Ticketing Systems in Public Transportation." In: *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*. WPES '13. New York, NY, USA: ACM, 2013, pp. 143–154. ISBN: 978-1-4503-2485-4. DOI: 10.1145/2517840.2517848. URL: http://doi.acm.org/10.1145/2517840.2517848 (visited on 02/17/2016).

[KO97] E. Kushilevitz and R. Ostrovsky. "Replication is not needed: single database, computationally-private information retrieval." In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc, 1997, pp. 364–373. ISBN: 0-8186-8197-7. DOI: 10.1109/SFCS.1997.646125. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=646125.

[KP13] Seny Kamara and Charalampos Papamanthou. *Parallel and Dynamic Searchable Symmetric Encryption*. Tech. rep. 2013/335. Microsoft Research, 2013, pp. 258–274. URL: https://eprint.iacr.org/2013/335 (visited on 04/28/2015).

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. "Dynamic searchable symmetric encryption." In: *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. Raleigh, NC, USA: ACM Press, Oct. 2012, pp. 965–976. ISBN: 978-1-4503-1651-4. DOI: 10.1145/2382196.2382298. URL: http://dl.acm.org/citation.cfm?id=2382196.2382298 (visited on 04/28/2015).

[KS05] Lea Kissner and Dawn Song. "Privacy-preserving set operations." In: *Advances in Cryptology–CRYPTO 2005* February 2005 (2005). URL: http://link.springer.com/chapter/10.1007/11535218_15.

[KS15] Martin Kroll and Simone Steinmetzer. "Automated Cryptanalysis of Bloom Filter Encryptions of Health Records." English. In: *Proceedings of the International Conference on Health Informatics*. Lisbon, portugal, 2015, pp. 5–13. ISBN: 978-989-758-068-0. DOI: 10.5220/0005176000050013. URL: http://www.scitepress.org/Portal/PublicationsDetail.aspx?ID=yygQ08Gqo8Q=&t=1 (visited on 06/09/2015).

[KSS09] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. "How to combine homomorphic encryption and garbled circuits." In: *Proceedings 1st International Workshop on Signal Processing in the EncryptEd Domain (SPEED'09)*. Vol. 2009. Sept. 2009, pp. 100–121. URL: http://www.trust.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_TRUST/PubsPDF/KSS09SPEED.pdf (visited on 06/09/2015).

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. "Predicate encryption supporting disjunctions, polynomial equations, and inner products." In: (Apr. 2008), pp. 146–162. ISSN: 3-540-78966-9, 978-3-540-78966-6. URL: http://dl.acm.org/citation.cfm?id=1788414.1788423.

[Kuz+11] Mehmet Kuzu et al. "A constraint satisfaction cryptanalysis of Bloom filters in private record linkage." English. In: *11th International Symposium on Privacy Enhancing Technologies (PETS 2011)*. Waterloo, ON, Canada: Springer Berlin Heidelberg, July 2011, pp. 226–245. ISBN: 978-3-642-22263-4. DOI: 10.1007/978-3-642-22263-4_13. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-22263-4_13 (visited on 04/17/2015).

[Kuz+12] Mehmet Kuzu et al. "A practical approach to achieve private medical record linkage in light of public resources." English. In: *Journal of the American Medical Informatics Association : JAMIA* 20.2 (July 2012), pp. 285–292. DOI: 10.1136/amiajnl-2012-000917. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3638181&tool=pmcentrez&rendertype=abstract (visited on 05/04/2015).

[KVC12] Alexandros Karakasidis, Vassilios S Verykios, and Peter Christen. "Fake Injection Strategies for Private Phonetic Matching." In: *Data Privacy Management and Autonomous Spontaneus Security*. Ed. by Joaquin Garcia-Alfaro et al. Vol. 7122. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 9–24. ISBN: 978-3-642-28878-4. DOI: 10.1007/978-3-642-28879-1. URL: http://link.springer.com/10.1007/978-3-642-28879-1.

[Lan01] Doug Laney. *Application Delivery Strategies*. Tech. rep. Feb. 2001. URL: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[LC11] Jaewoo Lee and Chris Clifton. "How much is enough? choosing epsilon for differential privacy." In: *Proceedings of the 14th international conference on Information security (ISC'11)*. Xi'an, China: Springer Berlin Heidelberg, Oct. 2011, pp. 325–340. ISBN: 978-3-642-24860-3. DOI: 10.1007/978-3-642-24861-0_22. URL: http://link.

springer.com/chapter/10.1007%2F978-3-642-24861-0_22 (visited on 03/20/2015).

[LCT14] T Lepoint, JS Coron, and Mehdi Tibouchi. "Scale-Invariant Fully-Homomorphic Encryption over the Integers." In: *17th International Conference on Practice and Theory in Public-Key Cryptography* 8383 (2014), pp. 311–328. URL: http://www.iacr.org/workshops/pkc2014/slides/6-1-scale-invariant-dghv.pdf.

[Len04] Arjen K. Lenstra. "Key lengths." In: *Handbook of Information Security* (2004), pp. 1–37. URL: ftp://outside.cs.bell-labs.com/cm/cs/who/akl/key_lengths.pdf.

[Lev66] Vladimir Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." In: *Soviet Physics Doklady* 10 (1966). Ed. by Paul E. Black, pp. 707–707. URL: http://xlinux.nist.gov/dads/HTML/Levenshtein.html.

[LH10] Heng Li and Nils Homer. "A survey of sequence alignment algorithms for next-generation sequencing." English. In: *Briefings in bioinformatics* 11.5 (Sept. 2010), pp. 473–483. DOI: 10.1093/bib/bbq015. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2943993&tool=pmcentrez&rendertype=abstract (visited on 06/09/2015).

[Li+10] Jin Li et al. "Fuzzy Keyword Search over Encrypted Data in Cloud Computing." In: *2010 Proceedings IEEE INFOCOM*. IEEE, Mar. 2010, pp. 1–5. ISBN: 978-1-4244-5836-3. DOI: 10.1109/INFCOM.2010.5462196. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5462196.

[Lin13] Yehuda Lindell. "Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries." English. In: *Proceedings 33rd Annual Cryptology Conference (CRYPTO 2013)*. Vol. 8043. LNCS. Santa Barbara, CA, USA: Springer Berlin Heidelberg, 2013, pp. 1–17. ISBN: 978-3-642-40084-1. DOI: 10.1007/978-3-642-40084-1_1. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-40084-1_1 (visited on 04/24/2015).

[Liu+11] Chang Liu et al. "Fuzzy keyword search on encrypted cloud storage data with small index." In: *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*. Beijing, China: IEEE, Sept. 2011, pp. 269–273. ISBN: 978-1-61284-203-5. DOI: 10.1109/CCIS.2011.6045073. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6045073 (visited on 04/28/2015).

[Liu+14] Chang Liu et al. "Search pattern leakage in searchable encryption: Attacks and new construction." In: *Information Sciences* 265 (2014), pp. 176–188. DOI: 10.1016/j.ins.2013.11.021. URL: http://www.sciencedirect.com/science/article/pii/S0020025513008293 (visited on 04/28/2015).

[LL04] Sven Laur and Helger Lipmaa. "On Private Similarity Search Protocols." In: *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*. 2004.

[LMA12] Yuri Lin, JB Michel, and EL Aiden. "Syntactic annotations for the google books ngram corpus." In: *Proceedings of the ACL ...* July (2012), pp. 169–174. URL: http://dl.acm.org/citation.cfm?id=2390499.

[LN14] T Lepoint and Michael Naehrig. *A Comparison of the Homomorphic Encryption Schemes FV and YASHE*. Tech. rep. 2014, pp. 1–18. URL: http://eprint.iacr.org/2014/062 (visited on 04/17/2015).

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey D Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014. ISBN: 978-1-107-07723-2. URL: http://www.mmds.org/#book.

[LWY07] Chen Li, Bin Wang, and Xiaochun Yang. "VGRAM: improving performance of approximate queries on string collections using variable-length grams." In: *Proceedings of the 33rd international conference on Very large data bases (VLDB)* (Sept. 2007), pp. 303–314. ISSN: 978-1-59593-649-3. URL: http://dl.acm.org/citation.cfm?id=1325851.1325889.

[Mar12] Marekich. *A set of four overlapping circles*. Sept. 2012. URL: http://commons.wikimedia.org/wiki/File:CirclesN4xb.svg (visited on 06/09/2015).

[McS10] Frank McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis." In: *Commun. ACM* 53.9 (Sept. 2010), pp. 89–97. ISSN: 0001-0782. DOI: 10.1145/1810891.1810916. URL: http://doi.acm.org/10.1145/1810891.1810916 (visited on 04/17/2015).

[Mel+09] Tamara Melnyk et al. "Linking Serial Sexual Assaults: The Taxonomic Similarity Index versus Jaccard's Coefficient." In: *American Psychology - Law Society*. San Antonio, TX, USA, Mar. 2009. URL: http://citation.allacademic.com/meta/p295927_index.html (visited on 06/11/2015).

[Mer90] Ralph C. Merkle. "A Certified Digital Signature." In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Vol. 435. 1990, pp. 218–238. ISBN: 978-0-387-97196-4. DOI: 10.1007/0-387-34799-2. URL: http://link.springer.com/chapter/10.1007/0-387-34805-0_21 (visited on 04/17/2015).

[Mic01] Michael t. Goodrich. *Efficient Authenticated Dictionaries with Skip Lists and Commutative Hashing*. Tech. rep. 2001.

[MNU05] Veli Mäkinen, Gonzalo Navarro, and Esko Ukkonen. "Transposition invariant string matching." In: *Journal of Algorithms* 56.Teollisuuskatu 23 (2005), pp. 124–153. ISSN: 0196-6774. DOI: 10.1016/j.jalgor.2004.07.008.

[Mul54] D E Muller. "Application of Boolean Algebra to Switching Circuit Design and Error Detection." In: *Transactions of the Institute of Radio Engineers* EC-3 (1954), pp. 6–12.

[Nag+13] Marcin Nagy et al. "Do I know you? - efficient and privacy-preserving common friend-finder protocols and applications." In: *Proceedings of the 29th Annual Computer Security Applications Conference on - ACSAC '13*. ACM Press, Dec. 2013, pp. 159–168. ISBN: 978-1-4503-2015-3. DOI: 10.1145/2523649.2523668. URL: http://dl.acm.org/citation.cfm?id=2523649.2523668 (visited on 04/21/2015).

[Nie+14] Frank Niedermeyer et al. *Cryptanalysis of basic Bloom Filters used for Privacy Preserving Record Linkage*. English. Tech. rep. Nürnberg: University of Duisburg-Essen, June 2014, pp. 1–21. URL: http://www.record-linkage.de/-download=wp-grlc-2014-04.pdf (visited on 06/09/2015).

[Niw+13] Suphakit Niwattanakul et al. "Using of Jaccard Coefficient for Keywords Similarity." English. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013*. Hong Kong, China: Newswood Limited, Mar. 2013, pp. 380–384. ISBN: 978-988-19251-8-3. URL: http://www.iaeng.org/publication/IMECS2013/ (visited on 05/27/2015).

[NN00] Moni Naor and Kobbi Nissim. "Certificate revocation and certificate update." In: *IEEE Journal on Selected Areas in Communications* 18.4 (2000), pp. 561–570. ISSN: 1-880446-92-8. DOI: 10.1109/49.839932.

[NP99] Moni Naor and Benny Pinkas. "Oblivious transfer and polynomial evaluation." In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. STOC '99. ACM, 1999, pp. 245–254. ISBN: 1-58113-067-8. DOI: 10.1145/301250.301312. URL: http://doi.acm.org/10.1145/301250.301312.

[NPG14] Muhammad Naveed, Manoj Prabhakaran, and Carl A Gunter. "Dynamic Searchable Encryption via Blind Storage." In: *Proceeding SP '14 Proceedings of the 2014 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE, May 2014, pp. 639–654. DOI: 10.1109/SP.2014.47. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6956592 (visited on 04/28/2015).

[NS98] David Naccache and Jacques Stern. "A new cryptosystem based on higher residues." In: *Proceedings of the 5th ACM conference on on computer and communication security* (1998), pp. 59–66. URL: http://dl.acm.org/citation.cfm?id=288106.

[Nyb93] Kaisa Nyberg. "Fast Accumulated Hashing." In: *Fast Software Encryption Lecture Notes in Computer Science*. 1993, pp. 83–87. URL: http://link.springer.com/chapter/10.1007%2F3-540-60865-6_45.

[OH04] H. Orman and P. Hoffman. *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*. RFC 3766 (Best Current Practice). Internet Engineering Task Force, Apr. 2004. URL: http://www.ietf.org/rfc/rfc3766.txt.

[OR07] Rafail Ostrovsky and Yuval Rabani. "Low distortion embeddings for edit distance." In: *Journal of the ACM* 54.5 (Oct. 2007), p. 23. DOI: 10.1145/1284320.1284322. URL: http://dl.acm.org/citation.cfm?id=1284320.1284322 (visited on 06/25/2015).

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. "A new public-key cryptosystem as secure as factoring." In: *Advances in Cryptology - EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 308–318. ISBN: 978-3-540-64518-4. DOI: 10.1007/BFb0054112. URL: http://link.springer.com/10.1007/BFb0054112.

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes." In: *Advances in Cryptography - Eurocrypt '99*. Lecture Notes in Computer Science 1592 (1999). Ed. by Jacques Stern, pp. 223–238. ISSN: 978-3-540-65889-4. DOI: 10.1007/3-540-48910-X. URL: http://www.springerlink.com/index/10.1007/3-540-48910-X.

[Pre+07] William H. Press et al. *Numerical Recipes - The Art of Scientific Computing (Third Edition)*. English. Cambridge, MA, USA: Cambridge University Press, 2007. ISBN: 978-0-511-33555-6. URL: http://numerical.recipes/aboutNR3book.html (visited on 09/29/2015).

[PSN10] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. "Cardinality estimation and dynamic length adaptation for Bloom filters." In: *Distributed and Parallel Databases* 28.2-3 (Sept. 2010), pp. 119–156. DOI: 10.1007/s10619-010-7067-2. URL: http://dl.acm.org/citation.cfm?id=1873101.1873113.

[Pun09] A. A. Puntambekar. *Software Engineering*. Technical Publications Pune, 2009. ISBN: 978-81-8431-559-2.

[Qui+89] Jean-Jacques Quisquater et al. "How to explain zero-knowledge protocols to your children." In: *Advances in cryptology - CRYPTO '89*. 1989, pp. 628–631. DOI: 10.1007/0-387-34805-0_60.

[Rab14] Tania Rabesandratana. *U.K.'s 100,000 Genomes Project gets £300 million to finish the job by 2017*. English. Aug. 2014. URL: http://news.sciencemag.org/biology/2014/08/u-k-s-100000-genomes-project-gets-300-million-finish-job-2017 (visited on 09/30/2015).

[Ran00] Kai Rannenberg. "Multilateral security a concept and examples for balanced security." In: *Proceedings of the 2000 New security Paradigms Workshop (NSPW 2000)*. 2000, pp. 151–162. ISBN: 1-58113-260-3. DOI: 10.1145/366173.366208.

[RB08] Matthew Robshaw and Olivier Billet, eds. *New Stream Cipher Designs - The eSTREAM Finalists*. 2008. URL: http://www.springer.com/computer/swe/book/978-3-540-68350-6.

[Ree54] I. Reed. "A class of multiple-error-correcting codes and the decoding scheme." In: *IRE Professional Group on Information Theory* 4 (1954), pp. 38–49. DOI: 10.1109/TIT.1954.1057465. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1057465 (visited on 04/17/2015).

[Roy+10] Indrajit Roy et al. "Airavat: security and privacy for MapReduce." In: *Proceeding NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation*. Berkeley, CA, USA: ACM Press, Apr. 2010, pp. 151–167. URL: http://dl.acm.org/citation.cfm?id=1855711.1855731.

[RS10] S Rane and W Sun. "Privacy preserving string comparisons based on Levenshtein distance." In: *IEEE WIFS 2010*. Dec. 2010. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5711449.

[RSA78] RL Rivest, A Shamir, and L Adleman. "A method for obtaining digital signatures and public-key cryptosystems." In: *Communications of the ACM* (1978). URL: http://dl.acm.org/citation.cfm?id=359342.

[Sal+14] Mohsen Amini Salehi et al. "RESeED: A Tool for Regular Expression Search over Encrypted Data in Cloud Storage." In: *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, May 2014, pp. 538–539. ISBN: 978-1-4799-2784-5. DOI: 10.1109/CCGrid.2014.58. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6846491 (visited on 04/28/2015).

[SB07] S Joshua Swamidass and Pierre Baldi. "Mathematical correction for fingerprint similarity measures to improve chemical retrieval." In: *Journal of chemical information and modeling* 47.3 (Jan. 2007), pp. 952–64. DOI: 10.1021/ci600526a. URL: http://dx.doi.org/10.1021/ci600526a.

[SBR09] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." In: *BMC medical informatics and decision making* 9.1 (Jan. 2009), pp. 41–41. DOI: 10.1186/1472-6947-9-41. URL: http://www.biomedcentral.com/1472-6947/9/41 (visited on 06/10/2015).

[SBR11] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. *A Novel Error-Tolerant Anonymous Linking Code*. English. Working Paper WP-GRLC-2011-02. Duisburg, Germany: University of Duisburg-Essen, 2011, pp. 2–14. URL: http://www.record-linkage.de/-download=wp-grlc-2011-02.pdf (visited on 06/09/2015).

[Sca+07] Monica Scannapieco et al. "Privacy preserving schema and data matching." In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07*. ACM Press, June 2007, pp. 653–653. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247553. URL: http://dl.acm.org/citation.cfm?id=1247480.1247553.

[SES14] Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. "Toward Practical Homomorphic Evaluation of Block Ciphers Using Prince." In: *2nd Workshop on Applied Homomorphic Cryptography and Encrypted Computing (WAHC 2014)* (2014).

[Sha49] Claude Shannon. "Communication Theory of Secrecy Systems." In: *Bell System Technical Journal* 28.4 (July 1949), pp. 656–715.

[Sha85] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes." In: *Advances in Cryptology*. Vol. 196. 1985, pp. 47–53. ISBN: 0-387-15658-5. DOI: 10.1007/3-540-39568-7_5.

[Sid15] Sidecar Technologies Inc. *Sidecar*. 2015. URL: http://www.side.cr/ (visited on 09/30/2015).

[SKS09] Meena Dilip Singh, P. Radha Krishna, and Ashutosh Saxena. "A privacy preserving Jaccard similarity function for mining encrypted data." English. In: *TENCON 2009 - 2009 IEEE Region 10 Conference*. IEEE, Nov. 2009, pp. 1–4. ISBN: 978-1-4244-4546-2. DOI: 10.1109/TENCON.2009.5395869. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5395869 (visited on 05/04/2015).

[SM02] Klaus U. Schulz and Stoyan Mihov. "Fast string correction with Levenshtein automata." In: *International Journal on Document Analysis and Recognition* 5.1 (Nov. 2002), pp. 67–85. DOI: 10.1007/s10032-002-0082-8. URL: http://link.springer.com/10.1007/s10032-002-0082-8.

[SPS14] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. "Practical Dynamic Searchable Encryption with Small Leakage." In: *Brieflings of NDSS Symposium 2014*. San Diego, California, USA, Feb. 2014, p. 15. URL: http://www.internetsociety.org/doc/practical-dynamic-searchable-encryption-small-leakage (visited on 04/28/2015).

[SS09] Yingpeng Sang and Hong Shen. "Efficient and secure protocols for privacy-preserving set operations." In: *ACM Transactions on Information and System Security ...* X.X (2009), pp. 1–34. URL: http://dl.acm.org/citation.cfm?id=1609965.

[SW05] Amit Sahai and Brent Waters. "Fuzzy Identity-Based Encryption." In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Springer Berlin Heidelberg, May 2005, pp. 457–473. ISBN: 978-3-540-25910-7. DOI: 10.1007/b136415. URL: http://dl.acm.org/citation.cfm?id=2154598.2154635.

[SW08] Elaine Shi and Brent Waters. "Delegating capabilities in predicate encryption systems." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5126 LNCS. 2008, pp. 560–578. ISBN: 3-540-70582-1. DOI: 10.1007/978-3-540-70583-3_46.

[SW81] Temple F. Smith and Michael S. Waterman. "Identification of common molecular subsequences." In: *Journal of molecular biology* 147.1 (Mar. 1981), pp. 195–7. URL: http://www.ncbi.nlm.nih.gov/pubmed/7265238.

[SWP00] Dawn Xiaodong Song, D. Wagner, and A. Perrig. "Practical techniques for searches on encrypted data." English. In: *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE Comput. Soc, 2000, pp. 44–55. ISBN: 0-7695-0665-8. DOI: 10.1109/

SECPRI.2000.848445. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=848445 (visited on 04/21/2015).

[Tam03] Roberto Tamassia. "Authenticated Data Structures." In: *Lecture Notes in Computer Science - Algorithms - ESA 2003*. Budapest, Hungary: Springer Berlin Heidelberg, 2003, pp. 2–5. ISBN: 978-3-540-39658-1. DOI: 10.1007/978-3-540-39658-1_2. URL: http://link.springer.com/chapter/10.1007%2F978-3-540-39658-1_2 (visited on 04/21/2015).

[TKC07] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. "Privacy preserving error resilient dna searching through oblivious automata." In: *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*. ACM Press, Oct. 2007, pp. 519–519. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315309. URL: http://dl.acm.org/citation.cfm?id=1315245.1315309.

[TS14] Stefan Tillich and Nigel Smart. *Circuits of Basic Functions Suitable For MPC and FHE*. 2014. URL: http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/ (visited on 09/30/2015).

[Ukk85] Esko Ukkonen. "Algorithms for Approximate String Matching." In: *nformation and Control*. 1985, pp. 19–19.

[VCV12] Dinusha Vatsalan, Peter Christen, and Vassilios S. Verykios. "A taxonomy of privacy-preserving record linkage techniques." In: *Information Systems* 38.6 (Nov. 2012), pp. 946–969. DOI: 10.1016/j.is.2012.11.005. URL: http://www.sciencedirect.com/science/article/pii/S0306437912001470 (visited on 04/23/2015).

[Vis+10] Bimal Viswanath et al. "An Analysis of Social Network-based Sybil Defenses." In: *Proceedings of the ACM SIGCOMM 2010 Conference*. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 363–374. ISBN: 978-1-4503-0201-2. DOI: 10.1145/1851182.1851226. URL: http://doi.acm.org/10.1145/1851182.1851226 (visited on 06/12/2015).

[Wan+09a] Rui Wang et al. "Privacy-preserving genomic computation through program specialization." In: *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*. ACM Press, Nov. 2009, pp. 338–338. ISBN: 978-1-60558-894-0. DOI: 10.1145/1653662.1653703. URL: http://dl.acm.org/citation.cfm?id=1653662.1653703.

[Wan+09b] R Wang et al. "Learning your identity and disease from research papers: information leaks in genome wide association study." In: *Proceedings of the 16th ACM conference on Computer and communications security*. ACM New York, 2009, pp. 534–544. ISBN: 978-1-60558-

894-0. DOI: 10.1145/1653662.1653726. URL: http://dl.acm.org/citation.cfm?doid=1653662.1653726 (visited on 05/27/2015).

[Wan+11] Cong Wang et al. "Harnessing the Cloud for Securely Solving Large-Scale Systems of Linear Equations." English. In: *2011 31st International Conference on Distributed Computing Systems*. Minneapolis, MN, USA: IEEE, June 2011, pp. 549–558. ISBN: 978-1-61284-384-1. DOI: 10.1109/ICDCS.2011.41. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5961757 (visited on 06/23/2015).

[Wan+14] Bing Wang et al. "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud." In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. IEEE, Apr. 2014, pp. 2112–2120. ISBN: 978-1-4799-3360-0. DOI: 10.1109/INFOCOM.2014.6848153. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6848153.

[WC09] WK Wong and DW Cheung. "Secure kNN computation on encrypted databases." In: *Proceedings of the 2009 ...* (2009). ISSN: 9781605585512. URL: http://dl.acm.org/citation.cfm?id=1559862.

[WEE14] WEELS Inc. *Bandwagon*. English. 2014. URL: http://bandwagon.io (visited on 09/30/2015).

[Wei13] Lei Wei. "Privacy-Preserving Regular Expression Evaluation on Encrypted Data." PhD thesis. 2013.

[Wet15] Kris Wetterstrand. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)*. English. June 2015. URL: http://www.genome.gov/sequencingcosts (visited on 09/30/2015).

[WWP08] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. "Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups." In: *Proceeding CANS '08 Proceedings of the 7th International Conference on Cryptology and Network Security*. Ed. by Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong. Vol. 5339. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Dec. 2008, pp. 178–195. ISBN: 978-3-540-89640-1. DOI: 10.1007/978-3-540-89641-8. URL: http://dl.acm.org/citation.cfm?id=1485310.1485327.

[WZ11] Wenling Wu and Lei Zhang. "LBlock: a lightweight block cipher." In: *Applied Cryptography and Network Security* (2011), pp. 327–344. URL: http://link.springer.com/chapter/10.1007/978-3-642-21554-4_19.

[Yao+04] Danfeng Yao et al. "ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption." In: (2004), pp. 1–32. ISSN: 1581139616. DOI: 10.1145/1030083.1030130.

[Yao86] Andrew Chi-Chih Yao. "How to generate and exchange secrets." In: *Foundations of Computer Science, 1986., 27th Annual Symposium on*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[Zer05] ZeroOne. *Mastermind*. Mar. 2005. URL: http://de.wikipedia.org/wiki/Datei:Mastermind.jpg (visited on 05/05/2015).

[ZLL13] Lan Zhang, Xiang-Yang Li, and Yunhao Liu. "Message in a Sealed Bottle: Privacy Preserving Friending in Social Networks." In: *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, July 2013, pp. 327–336. ISBN: 978-0-7695-5000-8. DOI: 10.1109/ICDCS.2013.38. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6681602.

[Zoo14] Zoosk Inc. *Zoosk*. English. 2014. URL: https://www.zoosk.com/ (visited on 09/30/2015).

[ZW14] Hongchao Zhou and Gregory Wornell. "Efficient homomorphic encryption on integer vectors and its applications." In: *2014 Information Theory and Applications Workshop (ITA)*. Ieee, Feb. 2014, pp. 1–9. ISBN: 978-1-4799-3589-5. DOI: 10.1109/ITA.2014.6804228. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6804228.